Diplomarbeit
Qualip: Software-Suite für die Qualitätskontrolle eines Linux-PCs

Andreas Claus

14. Januar 2003

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Einleitung

"Was ist Fortschritt?"

Einer der Gründer der Firma Intel, Gordon Moore, stellte 1965 bezüglich dieser Frage eine Regel auf, welche auch heute noch gilt: Laut "Moore's Law" verdoppelt sich die Anzahl der Transistoren auf einem Chip etwa alle 18 bis 24 Monate. Tatsächlich weisen die Benchmarks den PCs etwa eine Verdopplung der Rechenleistung innerhalb dieses Zeitraumes aus.

Jedoch bleibt die Beschleunigung der Programmausführung teilweise hinter den geweckten Erwartungen der Nutzer zurück, wenn ein Rechnersystem durch ein neueres ersetzt wird. Gründe dafür können neue Standards sein, die das Rechnersystem zwar erfüllt, die vom Betriebssystem jedoch nicht verwendet werden. Auch veraltete Treiber oder für den Nutzer unpassende Konfiguration der Hardware sind mögliche Ursachen. Der Nutzer kann sich mit seiner Einschätzung aber auch einfach irren.

Dass der Rechner die gegebenen Möglichkeiten zur Beschleunigung der Programmausführung nutzt[1], ist ein Indiz für die Qualität der Installation bzw. des PCs.

Bei der Auswahl der Komponenten eines Rechnersystems muss nicht nur auf deren generelle Verträglichkeit untereinander geachtet werden; die harmonische Abstimmung der Bestandteile aufeinander ist notwendig. Anderenfalls kann es zu einer Reduzierung der Geschwindigkeit oder Instabilitäten kommen.

Mit der höheren Taktfrequenz neuerer Hardware ist tendenziell auch der Stromverbrauch und damit die Abwärme der Rechnersysteme höher. Bei einem qualitativ gut konstruierten und installierten Rechner sollten diese Umstände auch bei einer Dauerbelastung nicht zu Instabilitäten führen.

Das Ziel dieser Arbeit ist die Schaffung einer Suite, welche die genannten qualitativen Aspekte eines PCs auf Basis des Betriebssystems Linux prüft.

Auf dem Markt der Betriebssysteme gewinnt Linux zunehmend an Bedeutung. Diese Entwicklung beschränkt sich nicht nur auf private Rechner. Linux ist auf dem Weg immer weiter auch in die kommerzielle Welt einzudringen. Firmen wie IBM bieten ihren Kunden bereits Server mit diesem Betriebssystem an. Auch in den Büros des Bundesdatenschutzbeauftragten wird in diesem Jahr auf Linux umgestellt.

Bei Linux handelt es sich um eine freie Software. Die im Quellcode verfügbaren Programme, welche für Linux angeboten werden, sind zahlreich und ebenfalls kostenlos zu erhalten.

Im ersten Kapitel dieser Ausarbeitung werden mittels Vorstellung der einzelnen Komponenten eines Rechnersystems, die zu testenden Eigenschaften festgelegt. Das Kapitel 2

[1] ohne gegebene Grenzwerte der Hersteller zu überschreiten (beispielsweise das Übertakten der CPU) oder die Funktionalität für andere Programme einzuschränken

stellt einige potentielle Testprogramme der Suite vor. Es wird erklärt ob, warum und wozu diese in der Qualip–Suite eingesetzt werden. Die vorgestellten Programme stellen dabei eine Vorauswahl dar. Nur die Programme, die eine Eignung für die Suite haben, werden tatsächlich behandelt. Es wird eine repräsentative Auswahl von Ergebnissen der Testläufe geschildert, durch die der Leser Schlussfolgerungen besser nachvollziehen kann. Diese Testergebnisse wurden teilweise durch die Programme selber, teilweise aber auch durch externe Geräte gewonnen. Kapitel 3 geht auf die einzelnen Tests der Suite ein. Es wird erklärt, auf welcher Grundlage diese ihre Wertung über die Qualität des Subsystems erheben.

Die Suite soll Administratoren beim Test neuer Rechnersysteme und Installationen oder bei der Eingrenzung von Fehlern unterstützen. Aber auch der “normale” Anwender kann Nutzen aus der Suite ziehen: beispielsweise um zu prüfen, ob sein subjektiver Eindruck zutrifft und das neu erworbenes Rechnersystem tatsächlich unter der Norm langsam arbeitet.

Notation

Es existieren unterschiedliche Hervorhebungen von Termini mit speziellen Bedeutungen:

- **prog** - Darstellung von Programmen

“*Datei*”

- - Darstellung von Dateinamen

<-opt>

- - Parameter, die Programmen übergeben werden
- **% Befehl** - Auf der Kommandozeile abgesetzte Anweisungen
- *Variable* - Hervorhebung von Wörtern oder Buchstaben mit besonderer Bedeutung wie Variablen
- [SITE-1] - Literaturangabe, welche auf eine Website im Internet verweist

Allgemeines

Wenn im Text Angaben zu Eigenschaften oder zur Existenz von Hardware oder Software gemacht werden, so beziehen sich diese immer auf den Zeitraum, in dem diese Arbeit entstanden ist.

Kapitel 1

Grundlagen und Anforderungen

1.1 PC

Die Definition des Begriffes PC änderte sich von 1980 bis heute, da durch die häufige Verwendung in der Umgangssprache immer mehr Rechnersysteme als PC bezeichnet wurden. PC ist die Abkürzung für Personal Computer oder auch IBM PC, was für ein Rechnersystem steht, welches auf einen Heimanwender zugeschnitten ist. So bezeichnete es im ursprünglichen Sinn ein Rechnersystem, was zu einem Zeitpunkt nur von einer Person genutzt werden kann. Im Rahmen dieser Ausarbeitung gilt jedes Rechnersystem als PC, auf dem Linux lauffähig ist und welches den PCI–Bus als primären Bus für die Anbindung von Erweiterungskarten nutzt. Diese Einschränkung ist nötig, um exotische Hardware wie Palmtops[1] oder Mainframes auszuschließen.

1.2 Linux

Bei Linux handelt es sich um den Kern eines Betriebssystems. Das Betriebssystem selber wurde von den Mitgliedern der Organisation GNU[2] entwickelt. Allerdings werden heute die Begriffe oft gemischt und der Kern mit GNU–Betriebssystem als Linux bezeichnet. Die Gründer der Software–Projekte waren Richard Stallman (GNU) und Linus Torvalds (Kernel). Das Ziel der Entwicklung von Linux war die Schaffung eines kostenlosen Unix-Derivats [Site-9]. Linux ist auf Rechnersystemen mit unterschiedlicher Architektur wie Alpha, Amiga, Atari, IBM, Intel, PowerPC/Macintosh, SGI, und Sun Sparc lauffähig.

Durch Distributoren wird Linux als Paket mit Anwendungen, Dokumentation und Programmen zur Systemeinstellung verkauft oder zum kostenlosen Download angeboten. Manche Distributoren, wie Suse, Red Hat, Caldera und Mandrake bieten außerdem einen kommerziellen Support an. Andere Distributoren, wie Debian und Slackware, halten mehr an dem ursprünglichen Konzept der freien Software fest [Site-10]. Die Qualip–Suite soll auf allen Distributionen ausgeführt werden können.

[1] Minicomputer in der Größe einer Handfläche, auch unter dem Namen Handheld bekannt

[2] `www.gnu.org`

1.3 Zu prüfende Eigenschaften

1.3.1 Schematische Darstellung der Komponenten

Bei den i386-kompatiblen Rechnersystemen stellen die North- und Southbridge, wie in Abbildung 1.1 zu erkennen, die Verbindung zwischen den Komponenten her. Somit werden die North- bzw. Southbridge durch ein Programm, das eine Interaktion zwischen zwei Komponenten bewirkt, ebenfalls beansprucht.

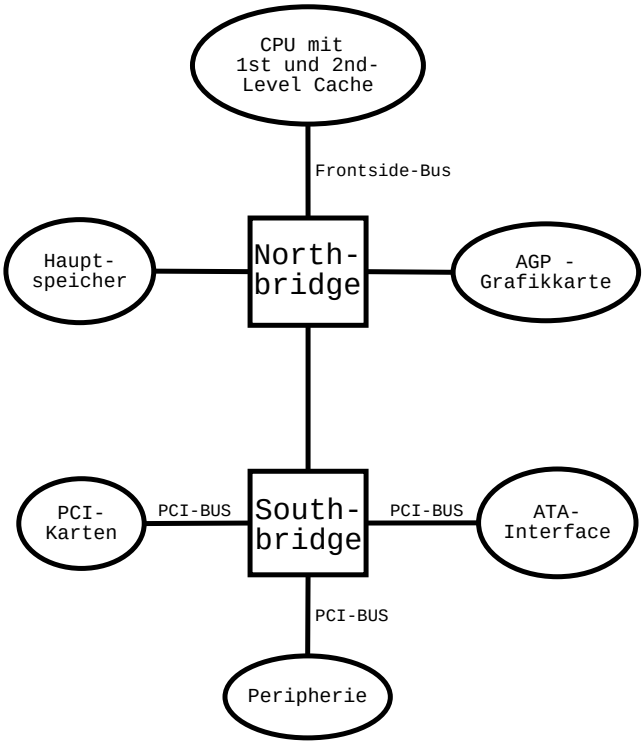


Abbildung 1.1: Verknüpfung der Komponenten eines i386-kompatiblen Rechnersystems

1.3.2 Die CPU

Die CPU hat die Aufgabe arithmetische und logische Befehle auszuführen. Für die Leistungsfähigkeit der CPU sind unter anderen folgende Komponenten entscheidend:

Taktfrequenz Diese reicht etwa von 300 bis 3000 MHz. Dabei wird versucht durch Verringerung der Strukturgröße (Verfeinerung des Fertigungsprozesses der CPU) immer höhere Taktfrequenzen zu erreichen. Durch Verkleinerung der Strukturgröße verringern sich die Verlustleistung und die Signallaufzeiten des Prozessors. Eine Erhöhung der Taktfrequenz führt dagegen zu einer größeren Verlustleistung [c't_07/2000].

Cachespeicher Bis auf wenige Ausnahmen[3] sind für die Datenpufferung zwischen Hauptspeicher und CPU zwei Cache-Ebenen vorhanden, der 1st- und der 2nd–Level–Ca-

[3] beispielsweise AMD K6-3 oder Server mit zusätzlichen Cache auf dem Motherboard

che. Sie dienen dazu, die Zugriffsgeschwindigkeit auf bestimmte Daten des Hauptspeichers zu beschleunigen. Der 1st–Level–Cache ist kleiner und verfügt über einen höheren Datendurchsatz als der 2nd–Level–Cache. Der 1st–Level–Cache gliedert sich in Befehls- und Daten–Cache.

Cache–Speicher sind im Vergleich zum Hauptspeicher wesentlich kleiner und schneller. In Tabelle 1.1 sind Angaben zu Größen und Datendurchsätzen von Cache- und Hauptspeicher verzeichnet. Dabei werden nur Rechner berücksichtigt, deren Architektur i386–kompatibel ist.

Speicherebene	min. Größe	max. Größe	Taktfrequenz
1st-Level-Cache	8 KByte[4]	64 KByte[4]	Prozessortakt
2nd-Level-Cache	64 KByte	2 MByte	Prozessortakt[5]
Hauptspeicher	32 MByte	4 GByte	100 - 533 MHz

Tabelle 1.1: Vergleich von Größe und Geschwindigkeit von Haupt- und Cache-Speicher auf i386–kompatiblen Rechnersystemen

Es gibt zeitliche und örtliche Cache-Strategien. Die zeitliche Strategie zieht Nutzen aus der Tatsache, dass auf Daten, auf welche ein Zugriff erfolgt ist, danach nochmals zugegriffen wird. Die örtliche Strategie zielt darauf ab, dass nicht nur auf ein einzelnes Datum, sondern auch auf dessen Nachbarn zugegriffen wird.

Wenn der Prozessor ein Datum aus dem Hauptspeicher anfordert, so wird zunächst überprüft, ob dieses sich schon im Cachespeicher befindet. Ist dies der Fall, so wird es aus dem schnelleren Cache und nicht aus dem Hauptspeicher geladen. Dies wird als *cache hit* (Treffer) bezeichnet. Befindet es sich nicht im Cache-Speicher, so wird dies *cache miss* (Fehlschlag) genannt.

Für die Trefferquote (Verhältnis von *cache hits* und Anfragen an den Cache) des Cache ist, neben den Programmen und deren Optimierung, die Assoziativität des Caches entscheidend. Sie schwankt bei i386–kompatiblen Prozessoren zwischen 4- und 16–facher Assoziativität [c't_03/2000].

Pipelines Die Anzahl der in der CPU vorhandenen Pipelines unterteilt sich in Integer- und Floating–Point–Pipelines. Dabei ist je nach Anwendung eine kürzere bzw. längere Pipeline besser geeignet. Durch die Verlängerung von Pipelines können die pro Stufe ausgeführten Berechnungen simpler (mit weniger Transistoren) gestaltet werden. Die Taktfrequenz kann erhöht werden. Jedoch sind bei Pipelines mit größerer Anzahl von Stufen mehr Takte für das Füllen und Leeren (etwa nachdem die Sprungvorhersage eine falsche Programmverzweigung prognostiziert hat) notwendig. Typische Werte für die Länge der Pipeline sind bei i386–kompatiblen Prozessoren zwischen 10 und 20 Stufen [c't_24/2000].

Frontside-Bus Dieser stellt die Verbindung zwischen CPU und Northbridge dar. Die Taktfrequenz des Frontside–Bus beträgt bei i386–kompatiblen Prozessoren zwi-

[4] bezieht sich nur auf den Datencache

[5] bis auf einige Ausnahmen mit einem Prozessortakt von unter 1 GHz, bei denen der 2nd–Level–Cache mit niedrigerer Taktfrequenz als die CPU angesprochen wird

schen 100 und 166 MHz. Dabei werden mit jedem Takt bis zu 4 Bits pro Pin[6] übertragen.

funktionale Einheiten Ein Prozessor kann über mehrere funktionale Einheiten verfügen. Mit diesen können Befehle eines Programms, die unabhängig voneinander ausführbar sind, parallel abgearbeitet werden. Sind mehrere Einheiten gleicher Funktion vorhanden, bezeichnet man den Prozessor als superskalar.

Registerzahl Je höher die Anzahl der Register ist, die ein Prozessor zur Verfügung hat, umso mehr Berechnungen können durchgeführt werden, ohne dass Daten mit den Caches ausgetauscht werden müssen. Dadurch ist es möglich, die für komplexe Berechnungen nötige Anzahl an Takten zu reduzieren.

Erweiterungen Heutige CPUs verfügen über Erweiterungen wie MMX, 3Dnow, SSE und SSE2. Sie wurden zur Beschleunigung von Multimedia–Anwendungen entworfen. Allen diesen Erweiterungen gemein ist die Vergrößerung des Befehlssatzes der CPU. Somit profitieren Programme nicht automatisch von diesen Erweiterungen.

64-Bit Architektur Durch den Übergang von 32- auf 64–Bit Architekturen mit Prozessoren wie dem Intel Itanium oder AMD Opteron werden Berechnung mit Variablen doppelter Genauigkeit mit niedriger Anzahl an Takten ausgeführt. Auch wird es mit den 64–Bit–Prozessoren möglich mehr Hauptspeicher zu verwalten (32–Bit–Prozessoren sind auf 4 GByte beschränkt) [c't_13/2002].

Je nach CPU, Mainboard und BIOS–Version lassen sich die Cache–Speicher der CPU einzeln deaktivieren und die Taktfrequenzen für Frontside–Bus und CPU verändern.

Ziel eines Tests ist es, die Leistungsfähigkeit der CPU in Relation zur Leistungsfähigkeit von anderen schon getesteten CPUs zu setzen. Somit können Rückschlüsse gezogen werden, ob eine CPU die für ihr Modell zu erwartende Leistung erbringt. Um die Geschwindigkeit einer CPU zu ermitteln, so dass sich ein praxisnaher Wert ergibt, sind diverse Tests notwendig, welche die Fähigkeiten der CPU in den einzelnen Disziplinen wie Integer- oder Floating–Point–Leistung testen müssen. Tests für Multiprozessor–Rechner werden in dieser Ausarbeitung nicht berücksichtigt. Bei Multiprozessorsystemen ist außer der Rechenleistung der einzelnen Prozessoren auch der Datendurchsatz zwischen den Prozessoren entscheidend. Es müsste der Datendurchsatz zwischen Prozessor und Hauptspeicher getestet werden, wenn beide Prozessoren den Speicher gleichzeitig zugreifen [c't_13/2002]. Dies würde den Rahmen dieser Arbeit sprengen.

1.3.3 Der Arbeitsspeicher

Im Arbeitsspeicher werden das Betriebssystem und Programme, welche vom Nutzer oder Betriebssystem gestartet wurden, gespeichert. Außerdem werden hier zu bearbeitende Daten gehalten. Schon seit über 20 Jahren wird hierfür der DRAM–Speicher (Dynamic Random Access Memory) eingesetzt. Es werden folgende DRAM–Speichermodule genutzt:

- SDR-SDRAM von 66 bis 133 MHz (Single–Data–Rate–Synchronous–DRAM)

[6] zur Datenübertragung bestimmter Pin

- DDR-SDRAM von 100 bis 200 MHz (Double–Data–Rate–SDRAM)
- RDRAM von 300 bis 533 MHz (DirectRambus–DRAM)

Die Geschwindigkeit der Speichermodule ist außer der Taktfrequenz durch etwa 30 weitere zeitbezogene Grenzwerte spezifiziert. In den Beschreibungen der Speichermodule finden sich aber meist nur drei dieser Grenzwerte.

RAS Precharge Time Zum Auslesen der Wertigkeit von Speicherzellen werden Leitungen auf eine bestimmte Spannung (den Referenzpegel) eingestellt und anschließend eine elektrische Verbindung zwischen diesen Leitungen und den Speicherzellen hergestellt. Die daraus resultierende Spannungsdifferenz wird durch Schreib- und Leseverstärker (Sense Amps) gemessen. In der "RAS Precharge Time" genannten Verzögerungszeit werden die Leitungen auf den Referenzpegel eingestellt.

RAS-to-CAS-Delay Die Zeitspanne, zwischen der Anforderung der Daten durch die Northbridge und Ermittlung des Speicherzelleninhalts durch die Sense Amps wird als "RAS-to-CAS-Delay" bezeichnet.

CAS Latency Während der "CAS Latency" werden die Daten aus den Sense Amps in die Ausgangstreiber[7] der SD–RAM–Chips[8] der Speichermodule übertragen.

Damit das Rechnersystem diese Grenzwerte selbstständig erkennen kann, sind auf den Speichermodulen zusätzliche Chips, SPD-EEPROMs genannt, vorhanden. Diese enthalten die Spezifikationen der Speichermodule. Jedoch werden die Angaben in den SPD-EEPROMs durch die Rechner teilweise nicht richtig erkannt. Gründe hierfür sind entweder ein nicht der Norm entsprechender SPD-EEPROM oder das BIOS des Rechners, welches die Werte nicht ausliest. Je nach Speicher, Mainboard und Version des BIOS sind diese Grenzwerte durch eine unterschiedliche Anzahl von Optionen des BIOS regelbar. Diverse Mainboard–Hersteller benennen diese Optionen allerdings um. Der Nutzer kann dann zwischen den Optionen "slow", "normal", "fast" und "turbo" wählen (teilweise ohne das diese Begriffe im Handbuch des Mainboards näher erläutert werden).

Bei Rechnern mit DDR–SDRAM–Speichermodulen ist die Einstellung "1T Command Rate" im BIOS der Rechner gebräuchlich. Mit dieser Einstellung wird die Northbridge angewiesen, Signale innerhalb eines Taktzyklus (des Speichertaktes) an die Speichermodule weiter zu reichen. Bei einer höheren Belastung der Adressleitungen durch Nutzung mehrerer Speichermodule ist dies meist nicht innerhalb eines Taktzyklus möglich [c't_08/2002].

Der Speicher ist zusätzlich zum Geschwindigkeitstest auch auf Fehler zu prüfen. Die Prüfung sollte Tests beinhalten, welche auch unbeständige Fehler des Speichers ermitteln.

[7] Verstärker, Schnittstelle zwischen der Innenschaltung und am Ausgang angeschlossenen integrierten Schaltungen, wandelt den Signalpegel von einer Innenschaltung zu einem Pegel um, mit dem am Ausgang angeschlossene Schaltungen arbeiten können, bei einem definiertem maximalen Laststrom

[8] es sind SD-RAM-Chips zwischen 16 und 512 MBit verfügbar, eine Gruppe zwischen 4 und 32 dieser Chips bilden ein Speichermodul

1.3.4 Die Grafikkarte

Derzeitig hergestellte Grafikkarten können sowohl zweidimensionale als auch dreidimensionale Objekte darstellen, dabei ist die Grafikleistung im 2D-Bereich in jedem denkbaren Anwendungsfall ausreichend. Zu überprüfen ist die 3D-Leistung der Grafikkarte, genauer ob OpenGL-Befehle schnell genug umgesetzt werden. OpenGL hat sich unter Linux als Standard für 3D-Anwendungen durchgesetzt.

Die zur Darstellung von 3D-Objekten nötige Rechenleistung gliedert sich in einen von der Grafikkarte und einen von der CPU berechneten Teil auf. Je nach Funktionsumfang des Prozessors der Grafikkarte übernimmt dieser mehr oder weniger der Rechenarbeit.

Gegenwärtig sind Grafikkarten üblich, welche die Transformations- und Beleuchtungsberechnung[9] durch ihren eigenen Prozessor erlauben. Dies sind beispielsweise die Familien der Geforce–Grafikchips der Firma Nvidia und Radeon–Grafikchips der Firma ATI. Mit Hilfe der T&L-Funktionen kann die Grafikkarte Objekte aus der Grundposition in alle möglichen Lagen und Größen transformieren. Sie berechnet für jeden Eckpunkt eines Dreiecks[10] die Helligkeit in Abhängigkeit von den vorhandenen Lichtquellen. Anschließend werden die Daten zur Anzeige auf dem Bildschirm in zweidimensionale Koordinaten umgewandelt und in die Bildschirmkoordinaten umgerechnet [c't_15/2002].

Jedoch muss zur Nutzung dieser Fähigkeiten ein Treiber verwendet werden, der diese bei Berechnungen einsetzt. Die OpenGL-Unterstützung der Grafikkarten bieten unter Linux unter anderem die Mesa-Bibliotheken (unter dem Namen glx oder xlibmesa). Es werden auch für einige spezielle Grafikkarten Treiber der Hersteller angeboten. Diese Grafikkarten sind unter anderem Matrox, ATI (Fire GL, Radeon 8500), Nvidia. Teilweise werden zu den Treibern auch OpenGL-Bibliotheken und Header–Dateien geliefert. Dies ist zum Beispiel bei Nvidia der Fall.

1.3.5 Das IO-Subsystem

Das IO-Subsystem ist für den Transport der Daten zwischen Festplatte und Hauptspeicher, sowie das Lesen und Schreiben der Daten verantwortlich. Für deren Datendurchsatz ist zum einen die Festplatte selber und zum anderen auch der Festplattencontroller verantwortlich. Um die mit dem Rechnersystem erreichbaren Datenraten zwischen Hauptspeicher und Festplatte zu realisieren, müssen Treiber und Einstellungen des Systems entsprechend gewählt sein. Der für den IDE- oder SCSI–Controller verantwortliche Treiber muss dessen Busmaster–Funktionalitäten[11] unterstützen. Ist dies gegeben, muss im Fall von IDE–Laufwerken eventuell noch deren Übertragungsmodus angepasst werden (Multiword–DMA, Ultra–DMA, usw.). Mit Hilfe eines Programms oder durch eine Voreinstellung des Herstellers kann der Cache einer Festplatte deaktiviert sein. Je nach Funktionalität der Festplatte kann ein Akustik–Management aktiviert werden, durch das die Festplatte die Lese- und Schreib–Köpfe langsamer bewegt. Das Deaktivieren des Caches und das Aktivieren des Akustik–Management verringert jeweils den Datendurchsatz der Festplatte.

[9] kurz T&L für Transform und Lighting

[10] Die Grafikkarte erhält ihre Daten in Form von Dreiecken bzw. Dreiecksnetzen, durch die die dreidimensionalen Oberflächen gebildet werden.

[11] dabei schreibt der Controller die angeforderten Daten direkt in den Hauptspeicher, ohne Beteiligung der CPU

Die Qualip–Suite soll alle im System befindlichen Festplatten auf deren Datendurchsatz bei Schreib- und Leseoperationen testen. Dabei ist zu prüfen, ob die Festplatten die für ihre Klasse bzw. Modell zu erwartenden Geschwindigkeiten erreichen. Da nicht davon ausgegangen werden kann, dass eine ungenutzte Partition für die Schreibtests zur Verfügung steht, muss in Kauf genommen werden, dass die Messung des Datendurchsatzes durch das verwendete Dateisystem beeinflusst wird.

Außer der Geschwindigkeit sind auch die SMART–Werte (Self–Monitoring, Analysis and Reporting Technology) der Festplatte zu untersuchen.

Die SMART–Werte bestehen aus aktuellen Werten und Limits. Bei einer intakten Festplatte sind die Zahlen der aktuellen Werte höher als die Limits. Die Limits können in zwei Gruppen von Härtegraden unterteilt werden. In der ersten Gruppe, Advisory, werden mögliche Fehler beschreiben, welche, falls sie zutreffen, schon aufgetreten sind. Die zweite Gruppe enthält Fehler, die immer mehr zunehmen und ab einem bestimmten Wert die Benutzung der Festplatte stören. Diese Gruppe wird PreFailure genannt. Werden Limits dieser Gruppe erreicht, so ist ein Ausfall der Festplatte in naher Zukunft wahrscheinlich [Site-14].

Die SMART–Werte enthalten Informationen über Fehler (wie “Bad Blocks”[12]) und Zustände (wie die Temperatur) der Festplatte. Jedoch ist die Eigenschaft bzw. der Zustand, worauf sich der jeweilige SMART–Wert bezieht, bei unterschiedlichen Herstellern bzw. Festplattentypen ein anderer. Somit kann ohne entsprechende Herstellerangaben einem SMART–Wert keine Eigenschaft (wie z.B. die Temperatur) der Festplatte zugeordnet werden.

IDE–Festplatten sind nicht für den Einsatz in Systemen gedacht, welche ununterbrochen in Betrieb sind. Jedoch werden sie auch für diesen Zweck auf Grund des günstigeren Preises (gegenüber SCSI–Festplatten, die die gleiche Datenmenge fassen) eingesetzt. SCSI–Festplatten haben einen höheren MTBF[13]- bzw. niedrigeren AFR[14]–Wert im Vergleich zu IDE–Festplatten (Erfahrungswert aus diversen Datenblättern). Die Wahrscheinlichkeit eines Ausfalls ist somit unter gleichen Bedingungen bei SCSI–Festplatten geringer als bei Festplatten mit IDE–Bus. Die MTBF–Werte berechnen sich bei IDE- und SCSI–Festplatten mit teilweise unterschiedlichen zugrunde liegenden Randbedingungen wie POH (Anzahl von Betriebsstunden innerhalb eines bestimmten Zeitraumes, typisch SCSI: 730.5 Stunden pro Monat (Dauerbetrieb), IDE: unterschiedlich, z.B. 333 Stunden pro Monat) und Duty Cycle (Prozentsatz der Betriebszeit, in der die Festplatte Positionier-, Schreib- und Lesezugriffen tätigt, typischer Wert SCSI: 30 %, IDE: 20 %) [Site-17].

1.3.6 Die Netzwerkverbindung

Zur Realisierung einer Netzwerkverbindung existieren die Techniken Ethernet, Fast- und GigaBit–Ethernet, ATM, WLAN (neben weiteren). Für ATM wurden allerdings nur experimentelle Treiber angeboten.

Es ist der Datendurchsatz der Netzwerkkarte bzw. der Netzwerkkarten zu überprüfen. Dazu ist ein Test zu wählen, welcher bei der Überprüfung des Datendurchsatzes nicht

[12] Blöcke der Festplatte, die auf Grund eines Defektes der Oberfläche nicht mehr genutzt werden können
[13] Mean Time Between Failure - die mittlere Zeitspanne (in Stunden) zwischen Fehlern
[14] Annual Failure Rate - die jährliche Fehlerwahrscheinlichkeit in Prozent

die Weiterverarbeitung der Pakete einer der OSI–Schichten größer als vier (Anwendungsschicht) mit in die Messung einfließen lässt. Die meisten[15] Anwendungen, welche Daten unter Linux über das Netzwerk übertragen, nutzen hierfür TCP oder UDP (beide Schicht vier des OSI–Referenzmodells). Wenn somit ein Programm den Datendurchsatz nur bis OSI–Schicht vier prüft, wird die allen diesen Anwendungsprogrammen zugrunde liegende Verbindung getestet.

Des weiteren ist auch ein Test auszuführen, welcher die Übertragung der Daten bis zur Anwendungsschicht (Schicht sieben) des OSI–Referenzmodells in der Messung erfasst.

Nach dem Test der Bandbreite ist zu prüfen, ob sich die Anzahl der fehlerhaften Frames, welche die Netzwerkkarte erreicht haben, erhöht hat.

1.3.7 Der Stabilitätstest

In einem Langzeittest (Laufzeit von 24h) ist die Stabilität eines Linux-Rechners zu prüfen. Instabilitäten können auftreten wegen:

1. eines Treibers, welcher Fehler aufweist,
2. einem Anwendungsprogramm, welches Fehler aufweist,
3. einem Teil der Hardware, welcher unzureichend gekühlt ist (betrifft Grafikkarte, CPU, Northbridge, Festplatte, spezielle Erweiterungskarten),
4. einem Netzteil, welches in bestimmten Situationen überlastet wird,
5. einem Spannungswandler auf dem Mainboard, welcher in bestimmten Situationen durch zu große Stromschwankungen oder Stromstärken überlastet wird (betrifft Grafikkarte und CPU),
6. durch einen Defekt der Hardware.

Bis auf Punkt zwei sind diese Ursachen zu testen. Jedoch wird es auf Grund der von Rechner zu Rechner unterschiedlichen Hardware nicht möglich sein, jede Hardware, welche im Rechner vorhanden ist, zu testen.

Besonders der Punkt drei, die Abführung der Wärme aus einem Rechnersystem, stellt ein immer größer werdendes Problem dar, das die Hersteller mit neuen Produkten zu lösen versuchen. So werden die Kühlkörper in den unterschiedlichsten Formen und Materialzusammensetzungen angeboten. Die Anzahl der Lüfter nimmt allgemein zu. Die Distributoren von Komplett-PCs müssen sich mit Luftströmungen im Gehäuse auseinander setzen. Dabei sind folgende zwei Fehler typisch. Die Abluft der Kühler strömt direkt auf andere Kühler zu und erhöht somit die Temperatur deren angesaugter Luft. Es werden zum Netzteil–Lüfter zusätzliche Gehäuse–Kühler im Rechnersystem installiert. Diese wirken auf Grund unbedachter Luftströmungen kontraproduktiv und Erhöhen somit die Temperatur gekühlter Komponenten des Rechnersystems.

Um Instabilitäten des Rechnersystems festzustellen müssen zwei Arten von Tests ausgeführt werden:

1. Tests, in denen die Komponenten des Rechners einzeln getestet werden

[15] außer Programme, welche einen eigenen TCP–Stack implementieren

2. Tests, in denen möglichst alle Komponenten des Rechners bestmöglich ausgelastet werden

Der erste Test widmet sich den Komponenten selbst, der zweite deren Zusammenwirken. Im zweiten Test wird somit auch die Stabilität des Netzteils geprüft. Durch die beiden Tests soll gesichert werden, dass auch sporadisch bzw. nur bei bestimmten Voraussetzungen auftretende Instabilitäten ermittelt werden.

Einige Prozessoren bzw. Mainboards unterstützen das Herabsetzen der Arbeitsgeschwindigkeit bei drohender Überhitzung. Diese Herabsetzung der Arbeitsgeschwindigkeit erfolgt durch Verringern der Taktfrequenz oder Einfügen von HLT- oder STPCLK–Befehlen. Diese Funktionalität ist z.B. in den CPUs Intel Pentium 4 der Firma Intel integriert. Mit Hilfe eines internen Temperatursensors überwacht sich der Prozessor selbständig und drosselt die interne Taktfrequenz bei bei drohender Überhitzung. Somit ist es nötig zu testen, ob ein Prozessor nach Ausführung von rechenintensiven Programmen noch die selbe Leistungsfähigkeit besitzt, wie nach einer Phase mit einer Prozessorauslastung von unter zehn Prozent.

Die Zeitschrift c't (Hans Heise Verlag) führt einen Stabilitätstest folgendermaßen durch: "Dabei laufen der 3DMark 2001, zwei SPEC-Benchmarks und zwei Kopieraktionen (von DVD und LAN-Laufwerk auf Festplatte) gleichzeitig ab und stressen somit einen Großteil des Systems." [c't_06/2002].

1.3.8 Die Softwareinstallation

Zur Qualität eines PC's zählen auch dessen Nutzungsmöglichkeiten bezüglich der darauf vorhandenen Programme. Somit muss geprüft werden, ob ein festzulegender Satz an Programmen auf dem Rechnersystem installiert ist.

Auf vielen der verbreiteten Linux-Distributionen ist ein Paket–Management vorhanden. Dies sind zum Beispiel Redhat, SUSE, Caldera, Mandrake und Debian. Das Paket–Management beinhaltet eine Datenbank im System, in der gespeichert wird, welche Software auf dem Rechner installiert ist. Auch werden in dieser Datenbank durch das Software–Paket installierte Dateien vermerkt, was bei einer eventuellen Deinstallation der Software genutzt wird.

Wenn vom Paket-Management unterstützt, ist weiterhin zu untersuchen, ob die Konfiguration der Pakete erfolgreich abgeschlossen wurde.

1.4 Allgemeine Fehler bei der Konfiguration der Hardware

Bei der Auswahl der Komponenten eines Rechnersystems muss auf die Verträglichkeit der Komponenten untereinander und auf eine harmonische Zusammenstellung der Komponenten geachtet werden. Anderenfalls kann es zu einer Reduzierung der Geschwindigkeit oder Instabilitäten kommen. Eine nicht harmonische Kombination von Komponenten ist beispielsweise die Nutzung einer CPU Pentium 4 der Firma Intel in Kombination mit SDR–SDRAM[16]. Der Prozessor profitiert von seinem hohen Frontside–Bus (siehe Abbil-

[16] Single-Data-Rate Synchronous Dynamic Access Memory, siehe Kapitel 1.3.3

dung 1.1) von 400 bzw. 533 MHz, während der Datendurchsatz zwischen Northbridge und Hauptspeicher zum “Flaschenhals” wird. Diese Konfiguration wird jedoch nicht durch die Qualip–Suite als Fehler identifiziert, da es sich dabei um eine aus diversen Gründen gewünschte Zusammenstellung der Komponenten handeln kann. Ein Beispiel für eine mangelnde Verträglichkeit ist die Southbridge 686B der Firma Via in Zusammenwirken mit der Soundkarte Soundblaster Live der Firma Creative.

Die Nutzung von PCI–Karten in Slots, welche sich den IRQ[17] teilen müssen und beide oft und gleichzeitig genutzt werden, kann zu einer verminderten Leistungsfähigkeit der PCI–Karten führen. Werden die beiden Komponenten, welche eine mangelnde Verträglichkeit aufweisen, bzw. die beiden PCI–Karten, die sich einen IRQ teilen, nicht in einem Test gleichzeitig beansprucht, so wird die daraus resultierende Instabilität bzw. verringerte Leistungsfähigkeit nicht durch die Qualip–Suite erkannt werden. Die Anzahl der möglichen Kombinationen von Komponenten und daraus resultierender notwendiger Tests ist zu groß.

1.5 Qualip–Suite

Die Qualip–Suite besteht aus Programmen, welche einen Test bzw. einen Testkomplex abarbeiten. Dabei soll jedes Programm einzeln aufrufbar sein. Die Programme müssen ohne Eingaben des Nutzers ablaufen. Es müssen Voraussetzungen geschaffen werden, damit sich neue Tests vergleichsweise einfach hinzugefügt lassen. Ein Programmierer muss demnach ohne lange Einarbeitungszeit neue Tests in die Suite einfügen können. Ein übergeordnetes Skript soll die Unterprogramme starten. Die Ausgabe der Unterprogramme muss einer einheitlichen Form folgen. Nach Möglichkeit sind Soll–Werte festzulegen, welche mit den erreichten Werten, unter Berücksichtigung der zugrunde liegenden Hardware, zu vergleichen sind. Somit ist es nötig die Hardware zu ermitteln und die Skalierungsfaktoren abzuleiten, um kommender Hardware gerecht zu werden.

[17] Anforderung der Hardware oder Software an die CPU die aktuelle Programmabarbeitung zu unterbrechen und einen bestimmten, dem IRQ zugeordneten Code auszuführen

Kapitel 2

Testprogramme

2.1 Arten von Tests

Programme, welche die Leistungsfähigkeit eines Systems ermitteln, unterteilen sich in die Gruppe der Benchmarks und der Quick–Hit–Tests. Die Gruppe der Benchmarks lässt sich in Synthetic und Application Benchmarks gliedern.

2.1.1 Quick-Hit-Tests

Quick-Hit-Tests sind Abfragen von Eigenschaften eines Rechnersystems, um einen bestimmten Aspekt, ein bestimmtes Detail der Leistungsfähigkeit eines Rechnersystems zu ermitteln. Ein Beispiel für einen solchen Test ist das Ermitteln der Taktfrequenz der CPU mittels Auslesen der Datei "*/proc/cpuinfo*". Diese Datei ist auf jedem Linux-System vorhanden[1]. Die ermittelte Taktfrequenz ist zwar ein Anhaltspunkt für die Leistungsfähigkeit eines Systems, doch schwankt die Leistungsfähigkeit zwischen Prozessoren gleicher Taktfrequenz aber unterschiedlicher Architektur teilweise recht stark (siehe dazu auch [SITE-5]). Weitere Beispiele für Quick-Hit-Tests sind die Programme **glxinfo** (Ausgabe von Parametern des OpenGL–Subsystems) und **mii-diag** (Testprogramm zum Prüfen und Einstellen der Konfiguration von Netzwerkkarten).

2.1.2 Synthetic Benchmarks

Synthetic (künstliche) Benchmarks prüfen ein, in seltenen Fällen auch mehrere Subsysteme eines Rechners gleichzeitig. Für dieses Subsystem ermitteln sie jedoch dessen Leistungsfähigkeit unter Nichtbeachtung der Leistungsfähigkeit der anderen Subsysteme und deren Zusammenwirken. Dabei wird das Subsystem präziser getestet als bei den Quick-Hit–Tests, indem ein Programm ausgeführt wird, das auf dieses Subsystem wirkt. Dabei wird die Ausführungszeit gemessen.

Die Programme testen ein Detail, oder eine Anzahl von Details des bzw. der Subsysteme. Diese Details lassen aber nur begrenzt Rückschlüsse auf die Leistungsfähigkeit des Subsystems in Anwendungsprogrammen zu. Ein Beispiel für eine solche Art von Benchmark ist das Programm **netperf**. Dieses Programm misst den Datendurchsatz einer

[1] Ausnahmen sind Rechner, bei welchen das "*/proc*"–Filesystem abgeschaltet wurde.

Peer-to-Peer Netzwerkverbindung. Es ist möglich, den maximalen Datendurchsatz einer TCP–Verbindung zu messen. Allerdings kann hieraus nicht die Datenrate einer NFS- oder FTP–Verbindung zwischen den beiden Rechnern bestimmt werden. Falls die Datenrate einer NFS- oder FTP–Verbindung geringer ist als erwartet[2], lässt sich durch **netperf** die Quelle eines eventuellen Fehlers einschränken.

2.1.3 Application Benchmarks

In der Klasse der Application (Anwendungs-) Benchmarks wird die Leistungsfähigkeit des Gesamt-Systems in so genannten “real-world” Anwendungen getestet. Dabei werden meist Teile des Codes gängiger Anwendungen genutzt, damit die ermittelte Leistungsfähigkeit möglichst genau mit der übereinstimmt, welche das System bei der Verwendung im Alltag hat.

So sind diese Art von Benchmarks Datenbank-Tests, Tests von Web- und Mail-Servern, Tests von Multiusersystemen, Tests von Anwenderprogrammen wie CAD, Berechnungen aus dem Wirtschaftsbereich, 3D-Modellierung. Application Benchmarks unter Linux sind beispielsweise Webstone, ein Benchmark für Webserver, und Postal, eine SMTP- und POP-Server Benchmark Suite. Der wohl geläufigste Anwendungsbenchmark unter Linux ist die Zeitmessung beim Kompilieren des Kernels [SITE-7] [SITE-6].

Die Synthetic Benchmarks eignen sich schlechter, um zu ermitteln wie leistungsfähig sich ein Rechnersystem in einem noch nicht bekannten Aufgabenbereich erweist.

Außerdem kann keine Aussage getroffen werden, wie stark die Leistungsfähigkeit eines Subsystems auf die des gesamten Rechnersystems einwirkt. So können z.B. bestimmte Prozessoren zwei Operationen gleichzeitig ausführen, wenn eine Floating-Point- und eine Integer-Operation zu bewältigen sind. Bei einem Synthetic Benchmark würde sich dieser Vorteil nicht offenbaren, wenn dieser die Floating-Point- und Integer-Rechenleistung getrennt ermittelt.

2.1.4 In der Qualip-Suite verwendete Arten von Benchmarks

Für die Qualip–Suite eignen sich sowohl Application- als auch Synthetic–Benchmarks. Es ist nicht bekannt, für welche Anwendung der Rechner genutzt wird. Dies spricht für die Nutzung der Application Benchmarks. Mit Synthetic Benchmarks können durch Anwendungen genutzte Subsysteme gezielt getestet werden. Da nicht alle Anwendungsprogramme getestet werden können, ist es somit nötig die Subsysteme in einem generellen Test zu prüfen.

Teilweise lassen sich Benchmarks nicht in die Gruppe der Synthetic- oder Anwendungs–Benchmarks einordnen, da sie Merkmale beider Gruppen aufweisen.

[2] einem Erfahrungswert, bezogen auf die verwendete Hardware

2.2 Die CPU

2.2.1 SPEC CPU2000

Die Organisation SPEC entwickelte den **SPEC CPU2000**-Benchmark [SITE-5], um die Rechenleistung der CPU auf möglichst vielen unterschiedlichen Rechnerarchitekturen vergleichen zu können. Um dieses Ziel zu realisieren, wird der Benchmark in Form von Quelltext geliefert, welcher aus Teilen von in der Praxis eingesetzten Anwendungenprogrammen besteht. Der Benchmark wird auf dem Zielsystem übersetzt. Er misst die Zeit für die Ausführung der einzelnen Unterprogramme und zieht daraus Schlüsse auf die Leistungsfähigkeit der CPU (Integer- und Floating-Point Operationen), der Speicher-Architektur (durch Variation des Datentyps und der Größe der Variablen) und des Compilers (Geschwindigkeitsmessung der Codeerzeugung, Geschwindigkeit des erzeugten Codes).

SPEC CPU2000 besteht aus 12 Einzeltests aus dem Bereich Integer-Berechnungen (Programmiersprachen C und C++) und 14 Floating-Point Tests (Programmiersprachen Fortran 77, Fortran 90 und C). Bei den Integer-Benchmarks wird die Zeit für die Ausführung der folgenden Tests[3] gemessen:

- das Komprimieren und Dekomprimieren von Daten
- das Übersetzen von C-Quelltexten
- das Berechnen von möglichen Zügen in einem Schachspiel
- die kombinatorische Optimierung von Logistik-Problemen

Die 14 Floating-Point-Tests enthalten unter anderem Quellcode aus folgenden Anwendungsgebieten:

- Programme zur Wettervorhersage
- Programme, welche Darstellungen von dreidimensionalen Objekten mit Hilfe der mesa-Bibliotheken berechnen
- einem Programm zur Erkennung von Gesichtern
- ein Programm, welches unter Nutzung eines Neuronalen Netzes aus Wärmebildern Objekte wie Flugzeuge filtern soll

Für die Qualip-Suite ist **SPEC CPU2000** jedoch weniger geeignet, da es einerseits nicht möglich war, es kostenlos zu erhalten und zum anderen **SPEC CPU2000** eine hohe Laufzeit hat, welche je nach Rechner bis zu mehreren Tagen betragen kann. Auch sind die Tests von der Leistungsfähigkit der CPU und des Hauptspeichers abhängig. Welchen Einfluss auf das Ergebnis eines Tests die Speicher-Architektur oder die CPU hat ist nicht erkennbar.

[3] Auszug

2.2.2 nbench

nbench basiert auf der Version 2 des BYTEmark–Benchmarks des Computermagazins BYTE. Der Programmcode dieses Benchmarks beinhaltet typische Algorithmen, welche in ähnlicher Form auch von in der Praxis eingesetzten Anwendungen genutzt werden. Die Laufzeit der einzelnen Tests des Benchmarks wird von diesem so gesteuert (etwa durch Einstellung der Schleifendurchläufe), dass die Gesamt–Laufzeit auf jedem Rechner etwa fünf Minuten beträgt. **nbench** vollführt nachfolgend beschriebene Tests.

2.2.2.1 Numeric Sort

Dieser Test prüft, wie schnell ein System ein nummerisches Feld (voreingestellt auf 8111 Elemente) sortieren kann. Das Feld ist eindimensional und die Zahlen sind Vorzeichen–behaftete 32-Bit Integer–Werte, wodurch bei 16-Bit Prozessoren die Geschwindigkeit nachhaltig beeinflusst wird. Das Sortierverfahren ist ein Heapsort-Algorithmus (mit einer Laufzeit von N $\log 2$ N). Dieses Sortierverfahren wird verwendet, weil dies ein in der Praxis oft genutzter Algorithmus ist. Es werden hauptsächlich 32-Bit–Integer–Vergleiche getestet. Der von **nbench** zurückgelieferte Wert entspricht der Anzahl von Feldern, die pro Sekunde sortiert werden konnten.

2.2.2.2 String Sort

Hiermit wird die Geschwindigkeit von Daten-Verschiebungen im Speicher gemessen. Dazu wird eine Folge von Datenblöcken von einem Teil des Speichers in einen anderen verschoben. Die Quell- und Ziel–Adressen sind dabei beliebig. Die Strings können eine Länge zwischen 4 und 80 Bytes aufweisen. Es werden immer Blöcke der Länge von acht Bit verschoben. Dieser Benchmark bedient sich ebenfalls des Heapsort–Algorithmus. Er arbeitet mit zwei Feldern. Ein Feld enthält die Strings selber, das andere enthält Offset–Adressen, welche auf die Strings im anderen Feld zeigen. Für jeden String im Feld gibt es einen Offset–Zeiger im anderen Feld. Dies ist nötig, da der Algorithmus die Strings auf Grund ihrer Index-Nummer schneller findet. Der zurückgelieferte Wert gibt die Anzahl der String–Felder an, die pro Sekunde sortiert wurden. Die Größe des String–Feldes ist auf einen Standardwert von 8111 Byte eingestellt.

2.2.2.3 Bitfield Operations

In diesem Benchmark wird getestet, wie schnell das System Bits im Speicher invertieren bzw. setzen kann. Die Bits verbleiben dabei an der gleichen Stelle. Der Benchmark simuliert eine Datenstruktur (genannt “bit map”), mit welcher in der Praxis die Speicherauslastung (beispielsweise einer Festplatte) verfolgt werden würde. Dabei steht jedes Bit für einen Block im Speicher und gibt dessen Belegung an. Ist ein solches Bit der Datenstruktur auf den Wert null gesetzt, so ist der entsprechende Block im Speicher frei. Bei einer Anforderung von freien Speicher zum Schreiben von Daten in den Speicher wird nach dem ersten Bit gesucht, das den Wert null hat. Dieses Bit wird invertiert und der entsprechende Block im Speicher beschrieben. Um dies in einem Benchmark zu simulieren, werden im Speicher zwei Blöcke reserviert. In einem Block wird die Datenstruktur

("bit map") angelegt, der andere Block enthält eine Reihe von Kommandos für die Datenstruktur ("bit map commands"). Jedes Kommando bewirkt in der Simulation, dass eine der folgenden drei Anweisungen ausgeführt wird:

1. eine Reihe von Bits auf den Wert null setzen
2. eine Reihe von Bits auf den Wert eins setzen
3. eine Reihe von Bits invertieren

Der "bit map command"–Block beinhaltet eine Gruppe von zufällig gewählten, der drei möglichen Kommandos. Jedes dieser Kommandos bezieht sich auf eine zufällige Anzahl von Bits. Während der Simulation wird jedes Kommando des Kommando–Blocks sequentiell ausgeführt. Dieser Benchmark liefert die Anzahl von Bits zurück, welche während der Simulation pro Sekunde bearbeitet werden konnten. Hierbei ist die Größe des "bit map"-Blocks konstant auf 128 KByte eingestellt, während die Größe des "bit map command"-Blocks je nach Prozessorgeschwindigkeit durch den Benchmark eingestellt wird.

2.2.2.4 Emulated Floating-Point

Die in diesem Benchmark enthaltenen Routinen sind ähnlich zu denen, welche ausgeführt werden, sobald das System Floating–Point–Werte berechnen muss und kein FPU–Coprozessor[4] vorhanden ist. Diese Routinen bestehen aus Befehlen, welche shift–Operationen, Integer–Additionen, Integer–Subtraktionen und Bit-Vergleiche enthalten. Der Benchmark bildet drei eindimensionale Felder, von denen die ersten zwei mit zufallsgenerierten Fließkommazahlen gefüllt sind und das dritte als Ergebnis–Speicher dient. Diese drei Felder sind dann weiter unterteilt in vier Gruppen. Jede dieser Gruppen ist eine der Grundrechenarten Addieren, Subtrahieren, Multiplizieren und Dividieren zugeteilt. Zum Beispiel wird für die Additions-Gruppe ein Element aus dem ersten Feld mit einem Element aus dem zweiten Feld addiert und das Ergebnis im dritten Feld gespeichert, wobei die Elemente jeweils den Additions-Gruppen der jeweiligen Felder entstammen. Dieser Benchmark liefert die Anzahl von Schleifen pro Sekunde zurück, wobei eine Schleife ein Durchlauf durch die Felder darstellt.

2.2.2.5 Fourier-Series

Mit diesem Benchmark wird die Leistung der FPU gemessen. Dies geschieht unter Anwendung von trigonometrischen[5] und transzendenten[6] Funktionen. Dazu werden die ersten n[7] Fourier–Koeffizienten der Funktion $(x+1)*x$ im Intervall von 0 bis 2 berechnet. Die Funktion $(x+1)*x$ wird als periodische Wellenform mit einer Periode von 2 behandelt. Wird eine Serie von Sinus- und Kosinus-Funktionen konstruiert und die

[4] Floating-Point-Unit, Teil der CPU, welcher Berechnungen mit Floating–Point–Werten direkt in Hardware ausführen kann

[5] Nutzung der Funktionen $\sin(x)$, $\cos(x)$, $\tan(x)$, $\cot(x)$

[6] Nutzung von Funktionen wie e^{ax}, $\sin(ax)$, $\sinh(x)$, $\ln(x)$

[7] Wert wird entsprechend gewählt um eine bestimmte Laufzeit zu erreichen

Fourier-Koeffizienten als Faktoren eingesetzt, so wird eine Annäherung an die originale Wellenform erreicht. Die Fourier-Koeffizienten werden dabei mittels Integration berechnet, was durch numerische Integration unter Anwendung der Trapezregel realisiert wird. Dieser Benchmark testet somit die Geschwindigkeit von den folgenden Floating-Point-Operationen: Berechnung von Sinus- und Kosinuswerten, Potenzieren, in geringerem Maß Floating-Point-Multiplikationen, -Divisionen, -Additionen und -Subtraktionen. Das Ergebnis des Benchmarks ist die Anzahl an Fourier-Koeffizienten, die pro Sekunde berechnet wurden. Dieses Ergebnis ist allerdings stark abhängig von den Mathematik-Bibliotheken, auf die der Compiler Zugriff hat. Falls dieser Benchmark auf gleichen Testrechnern mit verschiedenen Mathematik-Bibliotheken ausgeführt wird, kann es zu Abweichungen in den Resultaten kommen.

2.2.2.6 Assignment-Algorithm

Dieser Benchmark soll die Lösung eines in der Wirtschaft üblichen Problems nachstellen. Bei diesem Problem sollen mit Hilfe von X Maschinen Y Aufgaben gelöst werden. Jede der Maschinen kann jede Art der Aufgaben bewältigen. Jedoch sind die Kosten, die dabei entstehen, vom Maschine zu Maschine unterschiedlich. Auch die Kosten, welche eine bestimmte Maschine für verschiedene Aufgaben benötigt, unterscheiden sich. Weiterhin wird davon ausgegangen, dass eine Maschine nur eine Aufgabe bewältigt. Zur Lösung dieses Problems wird eine Matrix aufgebaut, in welcher die Kosten für die jeweilige Maschine und Aufgabe festgehalten sind. Ziel ist es, sämtliche Aufgaben bei minimalen Kosten zu lösen. Die Matrix ist ein zweidimensionales Feld von Long-Integer-Werten. Somit wird in diesem Benchmark getestet, wie gut das Rechnersystem Felder manipulieren kann. Es werden die minimalen Kosten auf Grundlage einer 101 x 101 Felder großen Matrix berechnet. Der vom Benchmark zurückgelieferte Wert gibt die Anzahl von Berechnungen der minimalen Kosten an, welche pro Sekunde ermittelt wurden.

2.2.2.7 Huffman Compression

Dies ist ein Kompressionsalgorithmus, welcher beispielsweise in Graphik-Dateiformaten Anwendung findet. Der Benchmark besteht aus drei Teilen:

1. Erstellen eines Huffman–Baums
2. Kompression
3. Dekompression

Ein Huffman–Baum ist eine spezielle Datenstruktur, welche die Kompression und die Dekompression leitet. Bei dieser Datenstruktur handelt es sich um einen binären Baum (jeder Knoten besitzt zwei Kindknoten). Zum Aufbau der Datenstruktur werden die unkomprimierten Daten eingelesen und eine Häufigkeitstabelle der enthaltenen Zeichen (ein Zeichen entspricht einem Byte Daten) erstellt. Auf Grund dieser Häufigkeitstabelle kann eine Code-Tabelle erzeugt werden. Dabei werden den am häufigsten vorkommenden Zeichen die Codes zugeordnet, welche am kleinsten bezüglich der Anzahl von Bits sind. Zur Komprimierung werden die Zeichen der Datenquelle nun durch den entsprechenden Code ersetzt, wodurch ein binärer String entsteht. Zur Dekomprimierung werden die Codes

aus dem binären String gelesen und mit Hilfe des Baumes durch die entsprechenden Zeichen ersetzt. Der Benchmark vollführt die Erstellung des Binärbaums, die Komprimierung und Dekomprimierung anhand eines unkomprimierten Textes mit einer Länge von 5000 Bytes. Es wird die Anzahl der Wiederholungen dieser drei Schritte pro Sekunde zurückgeliefert. Eine genauere Beschreibung des Algorithmus ist auf [SITE-1] zu finden. Für die durchgeführten Berechnungen werden hauptsächlich Integer–Operationen und Byte–Manipulationen genutzt.

2.2.2.8 IDEA-Encryption

Dieser Benchmark baut auf einen Verschlüsselungsalgorithmus auf. IDEA ist die Abkürzung von International Data Encryption Algorithm (international einsetzbarer Algorithmus zur Datenverschlüsselung). Der Algorithmus ist symmetrisch, was bedeutet, dass dieselbe Routine zum Ver- und Entschlüsseln genutzt wird. Er ver- und entschlüsselt die Daten in 64 Bit Blöcken. Dabei benutzt er drei Operationen:

1. xor
2. Addition modulo 216 (möglicher Überlauf wird ignoriert)
3. Multiplikation modulo 216+1 (möglicher Überlauf wird ignoriert)

Der Algorithmus benötigt einen Schlüssel mit einer Länge von 128 Bit. Der Schlüssel und die Datenblöcke werden weiter in 16-Bit Einheiten unterteilt, wodurch generell mit 16-Bit-Operationen gearbeitet wird. Der Benchmark arbeitet mit drei Datenpuffern: einem für den originalen Text, einem für den verschlüsselten Text und einem zum Schreiben des entschlüsselten Textes. Die Größe des zu verschlüsselnden Textes ist auf 4000 Bytes voreingestellt. Der vom Benchmark zurückgelieferte Wert gibt die Anzahl der Schleifen (wobei für eine Schleife einmal ver- und entschlüsselt werden muss) an, welche das System pro Sekunde durchlaufen hat.

2.2.2.9 Neural Net

Dieser Benchmark simuliert ein Neuronales Netzwerk, welches mit Rückwärts-Verkettung arbeitet. Das Netzwerk besteht aus drei Schichten. Es wird eine Eingabe von einem Bild simuliert. Dieses Bild besteht aus 5 x 7 Pixeln. Jeder Pixel entspricht einem Bit. Wenn das Bit gesetzt ist, ist der Pixel hell. Für einen dunklen Pixel hat das Bit den Wert null. Das Neuronale Netz soll das Bild interpretieren und den 8-Bit–ASCII–Code des Zeichens, welches es darstellt, zurückliefern. Die dazu ausgeführten Berechnungen sind primär das Potenzieren von Floating–Point–Variablen. Der Benchmark führt Lernzyklen in Schleifen aus. Ein Lernzyklus ist die vom Netzwerk benötigte Zeit, damit es lernt, den Eingangswerten die richtigen Ausgabewerte zuzuordnen, unter Beachtung einer gegebenen Toleranz. Der vom Benchmark zurückgelieferte Wert gibt die Anzahl der Lernzyklen pro Sekunde an.

2.2.2.10 LU-Decomposition

Dies ist ein Algorithmus zur Lösung linearer Gleichungssysteme (auch unter dem Namen LR-Zerlegung bekannt). Ausgehend von einer gegebenen Matrix $\mathbf{A}$, werden die Matrizen

$\mathbf{L}$ und $\mathbf{U}$ berechnet, sodass $\mathbf{L} \cdot \mathbf{U} = \mathbf{A}$ ist. Hierbei ist $\mathbf{L}$ eine normierte untere Dreiecksmatrix und $\mathbf{U}$ eine reguläre obere Dreiecksmatrix. Durch die Matrizen $\mathbf{L}$ und $\mathbf{U}$ wird das Lösen von linearen Gleichungssystemen der Form $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{b}}$ erleichtert. Außerdem lassen sich durch die LR-Zerlegung inverse Matrizen und Determinanten berechnen. Die im Benchmark verwendeten Algorithmen entstammen dem Buch [FlTeVe]. Der Benchmark erzeugt als erstes ein lösbares lineares Gleichungssystem. Dies geschieht, indem der Spaltenvektor $\vec{\mathbf{b}}$ mit ganzzahligen Zufallszahlen gefüllt wird und $\mathbf{A}$ mit einer Einheitsmatrix initialisiert wird. Die Gleichungen werden anschließend durch Multiplizieren der Zeilen mit einer Konstante und Addieren der Zeilen untereinander verändert. An diesem Gleichungssystem wird dann die LR-Zerlegung vollzogen. Von dem Benchmark wird die Anzahl von LR-Zerlegungen pro Sekunde zurückgeliefert, wobei die Größe der Matrix $\mathbf{A}$ 101 x 101 Elemente beträgt.

Der Test führt primär Additionen, Subtraktionen, Divisionen und Multiplikationen von Floating–Point–Werten aus.

2.2.2.11 Ausgabe der Ergebnisse

Es werden für jeden Test die erreichten Wiederholungen pro Sekunde ausgegeben. Zum Vergleich werden zu jedem Test Werte angegeben, die auf ein Rechnersystem mit AMD K6–CPU mit 233 MHz normiert sind. Auf diesem Vergleichs–Rechner wurde der Benchmark mit dem gcc–Compiler Version 2.7.2.3 und der libc–Bibliothek Version 5.4.38 übersetzt. Außer den Ergebnissen zu jedem Einzeltest werden auch Indices angegeben, in welchen die einzelnen Ergebnisse einfließen. Es wird ein Integer-, ein Floating–Point-, und ein Memory–Index gebildet. Diese sind ebenfalls wieder normierte Werte bezogen auf die Ergebnisse des Vergleichs-Rechners mit AMD K6-CPU mit 233 MHz. Die Indices bilden sich aus den Test–Ergebnissen wie folgt.

Integer-Index: Numeric Sort, Emulated Floating–Point, IDEA–Encryption, Huffman Compression

Floating-Point-Index: Fourier-Series, Neural Net, LU–Decomposition

Memory-Index: String Sort, Bitfield Operations, Assignment–Algorithm

2.2.2.12 Testergebnisse und Erläuterungen

Die zu verarbeiteten Datenmengen sind bei den in Tabelle 2.1 getesteten Rechnern kleiner als die Summe der Größen von 1st- und 2nd-Level-Cache. Dies lässt sich auch daran erkennen, dass die Testergebnisse zwischen gleichgetakteten AMD Athlon und Duron auf gleichem Level liegen[12]. Der für deren Leistungsfähigkeit maßgeblichste Unterschied

[8] Index–Werte normiert auf die Ergebnisse des Vergleichs-Rechners mit AMD K6-CPU, 233 MHz

[9] Wenn pro Pin und Takt mehr als ein Bit Daten übertragen werden, so wird statt der tatsächlichen die um diesen Faktor erhöhte Frequenz angegeben.

[10] Die SGI Origin verfügt über einen skalierbaren Crossbar an der 16 64-Bit-CPUs mit jeweils 64 KByte 1st-Level-Cache und 4 MByte 2nd-Level-Cache angeschlossen sind. Wird ein Programm gestartet, so wird es von dem Prozessor ausgeführt, welcher zu diesem Zeitpunkt am wenigsten ausgelastet ist.

[11] Die Bandbreite des System-Bus beträgt 5,12 GB/s.

[12] Zum Vergleich müssen aus der Tabelle 2.1 der AMD Athlon 1000 (Codename Thunderbird) und der AMD Duron 750 (Codename Spitfire) gewählt werden.

CPU-Hersteller und -Modell	Frontside-Bus (MHz)[9]	Speicher-takt (MHz)[9]	Integer-Index[8]	Floating-Point-[8] Index	Memory-Index[8]
AMD K6-2 500MHz	100	100	2.13	2.76	2.33
AMD Athlon 500MHz	200	100	2.42	5.61	2.66
AMD Duron 750MHz	200	133	3.69	8.44	4.09
AMD Athlon 1000MHz	200	133	4.93	11.26	5.39
AMD Athlon XP 1667MHz	266	266	8.31	16.09	8.96
AMD Athlon XP 1733MHz	266	333	8.60	16.71	9.27
Intel Pent.2 350MHz	100	100	1.48	2.66	1.53
Intel Pent.3 867MHz	133	133	3.70	7.35	3.97
Intel Pent.4 2000MHz	400	266	6.15	11.77	7.92
Intel Pent.4 2400MHz	533	333	7.43	15.24	9.47
SGI Origin 200MHz[10]	-[11]	- [11]	1.45	4.31	1.29

Tabelle 2.1: zu einem Rechner mit AMD K6 CPU (233 MHz) normierte Indices von **nbench** auf diversen Rechnersystemen

zwischen AMD Athlon (Codename Thunderbird bzw. Palomino) und Duron (Codename Spitfire bzw. Morgan) ist die Größe des 2nd-Level-Cache, welche beim Athlon 256 KByte und beim Duron 64 KByte beträgt [c't_13/2001]. Somit sind die Tests für die Speicherperformance der nbench-Suite für die Qualitätskontrolle eines PC's nur unzureichend. Es wird zusätzlich ein Programm benötigt, das den Datendurchsatz zwischen CPU und Hauptspeicher misst.

2.2.2.13 Begründung der Aufnahme in die Qualip-Suite

nbench fließt auf Grund von fünf Ursachen in die Testsuite ein:

- Die Tests orientieren sich an in Anwendungsprogrammen vorkommenden Befehlssequenzen, wodurch die Aussagekraft erhöht wird. Dabei ist die Laufzeit des Benchmarks mit etwa fünf Minuten recht kurz im Vergleich zum SPEC **CPU2000**–Benchmark (Kapitel 2.2.1)

- Da sich gezeigt hat, dass **nbench** die Leistungsfähigkeit der CPU realitätsnah zu charakterisieren vermag, hat sich der Benchmark zu einem gewissen Standard entwickelt. Er wird von diversen Computerzeitschriften genutzt.

- Bei der Übersetzung auf dem Zielsystem wird die Mathematik-Bibliothek hinzugelinkt. Falls diese falsch konfiguriert oder kompiliert wurde kann sich dies in einer langsameren Ausführungsgeschwindigkeit (doppelte Laufzeit und mehr) oder in einem Absturz dieser Programme äußern, welche diese Bibliothek nutzen.

- Aus den Laufzeiten werden Indices berechnet, welche die drei Grundcharakteristika eines CPU's darstellen: die Leistungsfähigkeit in den Bereichen Integer, Floating-Point und Speicherzugriff (wobei der ermittelte Index des Speicherzugriffs nur die Geschwindigkeit des Caches, nicht des Hauptspeichers widerspiegelt).

- Diese Indices beruhen jeweils auf mehr als einem Test. Liefert einer der Tests einen Wert zurück, der für diese Klasse von CPUs zu hoch oder tief ist (im Vergleich zu einem Prozessor, welcher in der Praxis ähnliche oder gleiche Leistungen erzielt), so wird der Index-Wert auf Grund der anderen einfließenden Tests nicht so stark beeinflusst, als wenn nur ein Test pro Index existieren würde.

2.3 Der Arbeitsspeicher

2.3.1 stream

Der **stream**-Benchmark testet die Dauer-Transferleistung zwischen CPU und Hauptspeicher. Dies wird durch Ausführen von Befehlen erreicht, welche mit Daten operieren, die größer als die Summe aller Cache-Speicher der CPU sind. Die Daten sind Vektoren, mit denen unterschiedliche Berechnungen durchgeführt werden. Die Vektoren werden bei den Berechnungen in Blöcke unterteilt, die der Prozessor verarbeiten kann. Somit wird die komplette Berechnung in mehreren Durchläufen ausgeführt. Da die verschiedenen Berechnungen eine unterschiedliche Belastung des Prozessors erzeugen, müssten sich die Transferraten ebenfalls unterscheiden. Wenn der Prozessor die Daten langsamer berechnet, als sie in den bzw. vom Hauptspeicher geschrieben bzw. gelesen werden können, so sinkt die Transferrate zum bzw. vom Hauptspeicher bei komplexeren Berechnungen. Dies stellt auch das erklärte Ziel des Benchmarks dar: Es wird ermittelt, ob der Hauptspeicher oder der Prozessor der begrenzende Faktor bei der Ausführung von Vektorrechnungen ist. Es werden demnach vier verschiedene Operationen mit den Vektoren durchgeführt:

1. `COPY:` $a(i) = b(i)$
2. `SCALE:` $a(i) = q * b(i)$
3. `SUM:` $a(i) = b(i) + c(i)$
4. `TRIAD:` $a(i) = b(i) + q * c(i)$

Die erste Operation belastet den Prozessor nur mit dem Berechnen von Adressen und dem Kopieren. Beim zweiten und dritten Test wird jeweils eine Floating–Point–Operation pro Iteration vom Prozessor geleistet. Bei Test vier sind es zwei Floating-Point-Berechnungen pro Iteration. Kann der Prozessor die Floating-Point-Berechnungen schneller durchführen, als die Daten vom Hauptspeicher gelesen bzw. geschrieben werden können, so sind die Werte für die vier Berechnungen annähernd gleich. Ist der Prozessor jedoch nicht in der Lage die Floating-Point-Berechnungen schneller durchführen, so sind die Werte des Datendurchsatzes des zweiten, dritten und vierten Tests niedriger als die des ersten. Handelt es sich bei der geprüften CPU nicht um einen superskalaren Prozessor, so benötigt dieser für den vierten Test mehr Takte als für Test drei. Bei Test zwei erfolgt im Vergleich zum dritten Test die Berechnung immer mit einem konstanten Wert, welcher während des Tests im Register oder Cache der CPU gehalten werden kann.

Es wurden mit **stream** Tests durchgeführt, deren Ergebnisse in Tabelle 2.2 verzeichnet sind.

Der Logik des Programmautors folgend ist es nicht nachvollziehbar, dass das der *COPY*-Test in 85 % aller Testläufe (in der Tabelle 2.2 wurden nicht alle durchgeführten Tests

CPU-Hersteller und -Modell	Hauptspeicher (MByte)	COPY MByte/s	SCALE MByte/s	ADD MByte/s	TRIAD MByte/s
SGI Origin 16 CPUs	6144	251	252	272	287
Intel Pent.4 2400 MHz	768 DDR-RAM	1233	1227	1448	1442
AMD Athl. XP 1400 MHz	768 DDR-RAM	684	700	834	711
AMD Athlon 850MHz	256 SD-RAM	316	318	345	342
AMD Athlon 500MHz	384 SD-RAM	377	360	469	400
AMD K6 233MHz	128 SD-RAM	87	87	90	89
Intel Pent.MMX 200MHz	64 EDO-RAM	86	86	91	91

Tabelle 2.2: die vier ermittelten Datendurchsätze von **stream** auf unterschiedlichen Rechnersystemen

angegeben) die langsamste und der *ADD*-Test die schnellste der vier Operationen ist. Der Grund hierfür ist die interne Berechnung der Datenrate. **stream** ermittelt diese durch Dividieren der übertragenen Daten durch die verstrichene Zeit. Bei den Tests *COPY* und *SCALE* wird ein Vektor geladen und ein Vektor gespeichert. Somit geht **stream** bei der zwischen Hauptspeicher und CPU übertragenen Datenmenge von dem doppelten der Datenmenge eines Vektors aus. Dagegen werden bei den Tests *ADD* und *TRIAD* zwei Vektoren geladen und einer gespeichert. Folglich rechnet **stream** bei den letzteren beiden Tests mit der 1.5-fachen Datenmenge im Vergleich zu den Tests *COPY* und *SCALE*.

Da die Größe der Vektoren mit 40 MByte die Größe der Cache–Speicher übersteigt, ist davon auszugehen, dass die Daten aus dem Hauptspeicher in den Cache geladen werden müssen (*cache miss*). Jedoch wurde bei bestimmten CPUs wie dem Intel Pentium 4 und AMD Athlon XP ein Hardware–Prefetch implementiert, wodurch es der CPU möglich ist, die Daten selbständig (parallel zu laufenden Berechnungen) in den Cache zu laden. Ist für die Verbindung zwischen CPU und Hauptspeicher der maximal mögliche Datendurchsatz erreicht, so kann auch der Prefetch der Daten die Abarbeitung von **stream** nicht beschleunigen.

Beim Speichern von Daten werden bei einem *cache miss* die Daten erst aus dem Hauptspeicher in den Cache geladen und dann wieder geschrieben (“Allocate on Write Miss Policy” [c’t_03/2000]). Dies ist notwendig, da der Cache immer nur mit kompletten Cache–Lines arbeitet und nicht über die Möglichkeit verfügt, eine beliebige Datenmenge zwischenzuspeichern. Somit werden mehr Daten zwischen Cache und Hauptspeicher übertragen, als in der Rechnung von **stream** eingehen. Die erzielten Datenraten liegen somit bei den Tests *COPY* und *SCALE* 50 %, bei *ADD* und *TRIAD* 33 % über den von **stream** ausgegebenen Datenraten. Die Cache–Lines, welche aus dem Hauptspeicher kopiert werden, können auch Daten enthalten, die nicht zu den zu verarbeitenden Vektoren gehören (dann müssten die genannten Prozentsätze erhöht werden). Somit kann die zwischen Hauptspeicher und CPU übertragene Datenmenge nicht exakt bestimmt werden. Jedoch ist davon auszugehen, dass jeder Benchmark, welcher den Durchsatz zwischen CPU und Hauptspeicher misst, die tatsächlich übertragene Datenmenge nicht bestimmen kann. Der Benchmark sollte aber auf einem Rechnersystem konstante Ergebnisse liefern und bei Hardware mit anderer Leistungsfähigkeit auch entsprechend unterschiedliche Ergebnisse ermitteln.

Bei nur 2-fach assoziativen Cache kann es[13] während der Tests *ADD* und *TRIAD* vorkommen, dass nur zwei der drei Vektoren (jeweils nur der Teil des Vektors der berechnet werden soll) zu einem Zeitpunkt im Cache stehen können [iX_07/1998]. Dies würde zu einer Verringerung des ermittelten Datendurchsatzes führen. Die Größe der Testvektoren lässt sich nur innerhalb des Quelltextes des Programms verstellen. Somit muss zur Nutzung von unterschiedlichen Größen der Testvektoren das Programm immer neu übersetzt werden. Der Test liefert Werte zurück, welche 10 % und mehr schwanken. Auf Grund dieser Probleme wird dieses Programm in der Qualip–Suite nicht eingesetzt.

2.3.2 memtest86

Das Programm **memtest86** wird mit einem eigenen Betriebssystem geliefert. Es kann somit nicht unter Linux ausgeführt werden, sondern muss über den Bootmanager bzw. wechselbare Datenträger gestartet werden. Wenn es gestartet wird, führt **memtest86** automatisch acht Tests aus. Diese acht Tests werden, sobald sie beendet sind, automatisch wieder von neuem gestartet. Der Nutzer kann das Programm jederzeit durch Betätigen der ESC–Taste beenden, wodurch ein Neustart des Rechners veranlasst wird. Über ein Konfigurationsmenü kann der CPU-Cache für alle Tests aktiviert oder deaktiviert werden. Es können die erweiterten Tests gewählt werden, **memtest86** führt dann vier weitere Tests aus. Weiterhin kann die Größe des Hauptspeichers von Hand eingestellt werden, falls sie von **memtest86** falsch erkannt wurde. Die Größe des Speichers wird über die BIOS–Funktion *e820* abgefragt. Alternativ kann **memtest86** die Größe des Speichers auch durch Zugriffs–Tests ermitteln. Im Konfigurationsmenü kann festgelegt werden, wie **memtest86** mit dem vom BIOS des Rechners reservierten Speicher umgeht. Die Standardeinstellung sieht vor, die reservierten Bereiche nicht zu testen.

2.3.2.1 Tests des Programms memtest86

Der ideale Test zum Ermitteln von sporadisch auftretenden Speicherfehlern ist folgendermaßen aufgebaut:

1. Eine Speicherzelle wird mit dem Wert 0 gefüllt.
2. Alle benachbarten Speicherzellen werden mit dem Wert 1 gefüllt. Dieser Schritt wird mehrfach wiederholt.
3. Es wird kontrolliert, ob die zuerst mit 0 gefüllte Speicherzelle immer noch den Wert 0 hat. hat

Mit diesem Test werden auch alle permanenten Fehler erkannt. Diese ideale Vorgehensweise ist aber nicht realisierbar, da sie ein genaues Wissen über die physische Lage der einzelnen Speicherzellen im Speicher erfordert. Außerdem ist der Aufbau der Chips von Hersteller zu Hersteller unterschiedlich. **memtest86** versucht sich durch zwei Algorithmen der idealen Vorgehensweise anzunähern. Der erste Algorithmus, *moving inversion* (wandernde Umkehrung) genannt, arbeitet wie folgt:

1. Der Speichers wird mit einem Bitmuster gefüllt.

[13] je nach Lage des Programms im Hauptspeicher

2. Es wird von der niederwertigsten Adresse aus gestartet und folgende drei Arbeitsschritte wiederholt ausgeführt.
   (a) Es wird geprüft, dass sich das Muster nicht verändert hat.
   (b) Das Muster wird erneut geschrieben, wobei jedes Bit des Musters vor dem Schreiben invertiert wird.
   (c) Die Adresse wird erhöht.
3. Es wird von der höchstwertigsten Adresse aus gestartet und folgende drei Arbeitsschritte wiederholt ausgeführt
   (a) Es wird geprüft, dass sich das Muster nicht verändert hat.
   (b) Das Muster wird erneut geschrieben, wobei jedes Bit des Musters vor dem Schreiben invertiert wird.
   (c) Die Adresse wird herabgesetzt.

Es können nur 4 bis 16 Bits auf einmal von einem Speicher–Chip gelesen oder geschrieben werden. Somit ist es unmöglich nur ein selektiertes Bit zu lesen oder zu schreiben. Es kann nicht garantiert werden, dass alle Speicherzellen auf ihre Wechselwirkung hin getestet wurden. Um dies zu kompensieren werden alle angrenzenden Zellen mit jeder möglichen Null- und Eins–Kombinationen geschrieben.

Der Speichertest kann durch Zwischenspeicherung im Cache oder anderen Pufferspeichern, sowie durch *out of order executions*[14] gestört und die Tests nutzlos werden. Wird beispielsweise ein Muster von 16 Bits in den Hauptspeicher geschrieben und anschließend sofort wieder gelesen, so wird es aus dem Cache gelesen. Es ist möglich den Cache der CPU abzuschalten, aber um auch alle Zwischenspeicher, welche zwischen CPU und Hauptspeicher angesiedelt sind, zu umgehen, wurde ein weiterer Algorithmus eingesetzt. Bei diesem Algorithmus, welcher Modulo-X genannt wird, werden folgende Schritte von der niedrigsten bis zur höchsten Adresse des Speichers, immer für einen Block von 20 Byte, ausgeführt:

1. In jedes 20. Byte wird ein Muster geschrieben.
2. In alle anderen Bytes wird das invertierte Muster geschrieben. Dieser Schritt wird mehrfach wiederholt.
3. Es wird überprüft, ob das Muster in jedem 20. Byte noch dem originalen Muster entspricht.

Durch die Anweisung 2 werden die Zwischenspeicher mit neuen Werten gefüllt, so dass in Anweisung 3 nicht aus diesen Zwischenspeichern gelesen werden kann.

memtest86 untersucht in insgesamt 12 Tests den Speicher auf Fehler. Diese Tests unterscheiden sich in dem verwendeten Algorithmus, im zu schreibenden Datenmuster und in den Cache–Einstellungen. Die Reihenfolge in der die Tests ausgeführt werden ist so gewählt, das Fehler so schnell wie möglich erkannt werden. Somit ist beim ersten Test

[14] wenn der Prozessor die Reihenfolge der Ausführung von Maschinenanweisungen ändert, um die für die Ausführung der Anweisungen benötigten Taktzyklen zu verringern

die Wahrscheinlichkeit am höchsten, beim letzten Test am niedrigsten, dass ein Fehler gefunden wird. Die letzten vier Tests werden nicht automatisch ausgeführt und müssen explizit durch die Option “extended testing” gewählt werden. Die Summe der Laufzeit dieser Tests ist etwa 13 mal[15] höher als die Summe der Laufzeit der ersten acht Tests. In den Tests wird folgendes realisiert:

Test 0: Es werden alle Adress–Bits in allen Speicherbänken getestet, indem bei Adresse Null begonnen wird und die Adresse immer um eins erhöht wird. Das Muster ist nach dem Schema *walking ones*[16] aufgebaut. Mit jeder Erhöhung der Adresse wird der Exponent der Zweierpotenz erhöht oder auf Null zurück gesetzt.

Test 1: Es wird der Algorithmus *moving inversion* genutzt. Dabei kommen Muster zum Einsatz, die vollständig mit dem Wert eins oder null gefüllt sind. Der Cache ist aktiviert, wodurch der Test einerseits ungenauer, andererseits aber auch schneller abläuft als mit deaktiviertem Cache.

Test 2: In jede Adresse wird der Adresswert selber geschrieben. Anschließend wird die Beständigkeit der geschriebenen Werte geprüft. Der Cache ist in diesem Test deaktiviert.

Test 3: Dieser Test ist ähnlich dem Test 1, allerdings wird hier mit einem 8 Bit großen Muster gearbeitet, welches nach dem Schema *walking ones* oder *walking zeros*[17] aufgebaut ist. Der Cache ist aktiviert.

Test 4: Dieser Test arbeitet mit dem *moving inversion* Algorithmus und 32–Bit–großem Muster bei aktiviertem Cache. Das Muster wird bei jedem untersuchten Speicherblock nach links geshiftet. Die Adresse, welche als nächste beschrieben wird, ergibt sich, indem die Adresse des vorherigen Durchlaufs nach links geshiftet wird.

Test 5: Dieser Test verursacht eine hohe Last, indem er Befehle nutzt, welche Blöcke im Speicher verschieben können (mit Hilfe des Assembler–Befehls *movsl*). Der Test wurde auf Basis des **burnBX**–Programms[18], welches von Robert Redelmeier geschrieben wurde, entwickelt. Es wird ein 64 Bit großes Muster genutzt, welches abwechselnd invertiert und nicht invertiert in den Speicher geschrieben wird. Anschließend werden 4 MByte große Teile des Speichers mit Hilfe des *movsl*–Befehls 64 mal verschoben. Es folgt ein Test auf Übereinstimmung mit dem Muster. Ergibt dieser Test einen Fehler, ist es nicht möglich die genaue Stelle zu bestimmen, an der der Fehler seinen Ursprung hat, da unbekannt ist, nach welcher Verschiebung der Fehler aufgetreten ist. Da die Verschiebungen aber innerhalb eines 8 MByte

[15] schwankend siehe Kapitel 2.3.2.2

[16] Zweierpotenzen - Binärziffern enthalten eine einzige Eins, welche mit jedem Schritt weiter rückt oder wieder bei Stelle null beginnt

[17] Alle Binärziffern sind mit Einsen gefüllt außer einer einzigen Null, welche mit jedem Schritt zur nächsten oder ersten Binärziffer im Block rückt.

[18] Beschreibung in Kapitel 2.9.4

Blocks realisiert werden müssen, lässt sich die Fehlerstelle begrenzen. Der Cache bleibt während des Tests aktiviert.

Test 6: In diesem Test werden, gleich Test 1, nur Muster verwendet, welche entweder komplett mit dem Wert eins oder null gefüllt sind. Jedoch arbeitet dieser Test mit dem *modulo-X*–Algorithmus. Somit werden Fehler gefunden, welche auf Grund von Zwischenspeichern bei den vorherigen Tests nicht erkannt wurden.

Test 7: Dieser Test unterscheidet sich zu Test 1 nur in dem Punkt, dass der Cache deaktiviert ist. Der Test ist somit langsamer als Test 1.

Test 8: Bei diesem Test wird der Test 5 wiederholt, nur dass die Anzahl der Verschiebungen im Speicher von 64 auf 512 erhöht wird.

Test 9: Der Algorithmus *moving inversions* kommt bei diesem Test zur Anwendung. Es wird ein 8 Bit großes Muster genutzt und der Cache ist deaktiviert.

Test 10: Dieser Test arbeitet mit dem *modulo-X*–Algorithmus und aktiviertem Cache. Es wird jedoch im Gegensatz zu Test 6 nicht mit Mustern gearbeitet, welche nur mit Nullen oder Einsen gefüllt sind. Der Inhalt der Muster variiert während des Tests. Es handelt sich um ein 8 Bit Muster.

Test 11: Der letzte Test arbeitet mit einem 32 Bit Muster, welches ebenfalls variiert. Der Cache ist deaktiviert und der Algorithmus *moving inversions* wird verwendet.

2.3.2.2 Laufzeit von memtest86

memtest86 belegt die ersten 91 KByte des Hauptspeichers, diese werden somit wahrscheinlich[19] nicht getestet. Andere Programme wie **ctRAMtst** umgehen dies dadurch, dass sie sich in den Speicher der Grafikkarte schreiben. Jedoch weisen die Programmierer von **ctRAMtst** darauf hin, dass der Test dadurch langsamer ausgeführt wird. Außerdem bringt **ctRAMtst** kein eigenes Betriebssystem mit.

Die Ausführungszeiten des Programms **memtest86** auf einer Auswahl unterschiedlicher Rechner können in Tabelle 2.3 eingesehen werden (bei einer Toleranz von etwa 5 %).

[19] Die Dokumentation enthält keinen Hinweis hierzu.

	Intel Celeron 366 MHz 64 MByte SD-RAM	AMD Athlon XP 1733 MHz[20] 756 MByte PC2700-RAM	Intel Pentium 4 2400MHz[20] 1024 MByte PC800-RAM
Test 0	5 s	3 s	0.3 s
Test 1	18 s	31 s	18 s
Test 2	1 m 2 s	1 m 30 s	4 s
Test 3	1 m 38 s	2 m 49 s	1m 48 s
Test 4	8 m 5 s	14 m 54 s	7 m 45 s
Test 5	1 m 40 s	3 m 25 s	2 m 24 s
Test 6	4 m 24 s	7 m 54 s	4 m 49 s
Test 7	6 m 4 s	8 m 39 s	36 s
Standardtests Summe	23 m 16 s	39 m 44 s	17m 44.3s
Test 8	12 m 30 s	27 m 8 s	18 m 24 s
Test 9	49 m 30 s	1 h 20 m 27 s	4 m 49 s
Test 10	30 m 34 s	33 m 21 s	23 m 31 s
Test 11	3 h 29 m 40 s	9h 16 m 7s	1 h 36 m 28 s
Summe aller Tests	5 h 25 m 30 s	12 h 6 m 51 s	2 h 40 m 56.3 s

Tabelle 2.3: Ausführungszeiten des Programms **memtest86** auf unterschiedlichen Rechnersystemen

2.3.2.3 Begründung der Aufnahme in die Qualip-Suite

Dieser Test liefert sein eigenes Betriebssystem mit. Diese Eigenschaft erlaubt es **memtest86** Adressen im Speicher direkt anzusprechen. Linux arbeitet mit virtuellem Speicher und verwaltet diesen für alle Programme. Auch ist es einem Speichertestprogramm nicht möglich Speicherbereiche zu testen, welche von Linux selber, oder von anderen Programmen in Benutzung sind. Somit ist die Aussagekraft eines Speichertestprogramms, welches von Linux aus gestartet wird, nur gering. Viele Speichertestprogramme setzen auf DOS (z.b. **ctRAMtst**) auf, da dies ein *"real mode"*–Betriebssystem ist und das Programm somit den Speicher direkt ansprechen kann. Jedoch muss sich hierfür ein startbares DOS auf der Festplatte bzw. anderen bootbaren Medium befinden. Bis auf **memtest86** brachte keines der im Laufe dieser Arbeit untersuchten Speichertestprogramme ein eigenes Betriebssystem mit.

2.3.3 cachebench

2.3.3.1 Überblick

cachebench ist ein Benchmark, welcher das Leistungsverhalten des Speichersubsystems bewertet. Das Ziel ist es, die Größe und Geschwindigkeit der einzelnen Caches–

[20] 133 MHz Frontside-Bus

Ebenen und des Hauptspeichers zu ermitteln. Es wird untersucht, wie wirksam die Cache–Strategie[21] ist, wobei die Größe der zu verarbeiteten Daten schrittweise erhöht wird. Bei entsprechender Größe der Daten ist eine Zwischenspeicherung der kompletten Daten im Cache nicht mehr möglich, so dass die Geschwindigkeit des Hauptspeichers gemessen wird.

cachebench umfasst zur Zeit acht verschiedene Einzelbenchmarks. Jeder Benchmark greift dabei wiederholt auf Daten mit variierender Vektorlänge zu. Dabei wird für jede Vektorlänge die Zahl von Wiederholungen für eine gegebene Zeit gemessen. Um die gesamte Datenmenge zu berechnen, auf welche zugegriffen wurde, wird das Produkt aus Vektorlänge und Anzahl der Wiederholungen gebildet. Wird die Datenmenge durch die gesamte Laufzeit geteilt, ist das Ergebnis die Bandbreite des Speichers. Die Bandbreite wird in MByte/s ausgegeben. Die acht verschiedenen Speichertests des Benchmarks sind:

1. Read (lesen),
2. Write (schreiben),
3. Read/Modify/Write (lesen, ändern, zurückschreiben),
4. handtuned Read (handoptimiertes lesen),
5. handtuned Write (handoptimiertes schreiben),
6. handtuned Read/Modify/Write (handoptimiertes lesen, ändern, zurückschreiben),
7. memset() (memset-Funktion der C-Bibliothek),
8. memcpy() (memcpy-Funktion der C-Bibliothek).

Die ersten drei Tests sind in Bezug auf die Komplexität des Quelltextes einfacher aufgebaut als die restlichen fünf Tests. Sie dienen zum Vergleich, wie gut der Compiler arbeitet. Wenn dieser effektiv arbeitet, sind die erzielten Werte ähnlich denen, welche beim entsprechenden handoptimierten Test erreicht werden. Die beiden Tests sieben und acht sind als Anhaltspunkte für Vergleiche gedacht. In diesen beiden Tests werden die Funktionen memcpy() und memset() der C–Bibliothek genutzt, welche in mit C programmierten Anwendungsprogrammen oft genutzt werden. Dabei werden keine Berechnungen durchgeführt, sondern nur Speicherbereiche mit Werten gefüllt, welche entweder vorgegeben sind oder von anderer Stelle kopiert werden.

Die ersten sechs Tests rechnen mit Daten aus Feldern eines vordefinierten Datentyps. Dieser wird zur Übersetzungszeit festgelegt. Der Standardwert ist der Datentyp Double. Es können alternativ Char, Int und Float verwendet werden. Die Datentypen unterscheiden sich neben weiteren Merkmalen auch in der Größe des Speichers, den sie belegen. Eigenschaften der Prozessoren wie die Größe der Register führen zu verschiedenen Ergebnissen in Zusammenhang mit dem jeweils verwendeten Datentyp.

[21] das Prefetching der Daten - kann durch den Programmierer (SSE-Erweiterungen ab Intel Pentium 3 und 3Dnow!-Erweiterungen ab AMD K6-2, Athlon) oder Compiler vorgegeben werden, funktioniert aber auch automatisch bei AMD Athlon XP und Intel Pentium 4 durch den Prozessor, ist dann allerdings weniger effektiv [c't 24/2000]

Alle Benchmarks haben eine vor dem Programmstart festzulegende Laufzeit. Die Laufzeit ist für alle auf Grund dieses Programmaufrufs ausgeführten Tests gleich. Der Vorteil dieser Lösung liegt darin, dass der Test auf Systemen unterschiedlicher Geschwindigkeit die gleiche Laufzeit hat und damit Ungenauigkeiten durch zu kurze Laufzeiten vermieden werden können.

2.3.3.2 Cache Read

Dieser Test prüft die Geschwindigkeit für compileroptimierte Lesezugriffe auf Vektoren verschiedener Größen. Für den Fall, dass die Größe der Vektoren kleiner als die Cachegröße ist, werden die Daten aus dem Cache geladen. Der Zugriff auf die Daten wird dadurch beschleunigt. Der Pseudocode für dieses Teilprogramm sieht folgendermaßen aus:

```
for all vector length // wiederhole folgende Anweisungen für jede Vektorlänge
  timer start
  for iteration count // wiederhole für die vorgegebene Anzahl folgende Anweisungen
    for I = 0 to vector length // für alle Datenblöcke bis zur aktuellen Vektorlänge
      register += memory[I] // lies einen Datenblock und addiere ihn zu register
  timer stop
```

Die Variable *register* ist von einem vordefinierten Typ, welcher Integer, Float, Double oder Char sein kann. Das Feld *memory* ist mit null-Werten gefüllt. Die Variablen *I*, *register* und der Zeiger auf das Feld *memory* sind, wenn der Compilerschalter REGISTER gesetzt ist, als register-Variablen vereinbart. Die register-Vereinbarung von Variablen in der Programmiersprache C bedeutet, dass die Variablen in den Registern des Prozessors gehalten werden sollen. Ist dies aus Mangel an Anzahl oder Größe der Register nicht möglich, so werden sie im Hauptspeicher bzw. Cache gehalten [KerRit]. Der Compilerschalter REGISTER wird gesetzt, indem der Compiler mit der entsprechenden Option aufgerufen wird. Bei dem GNU C-Compiler gcc wird dies durch die Option “-DREGISTER“ erreicht.

2.3.3.3 Cache Write

Dieser Benchmark liefert die Schreibgeschwindigkeit für vorgegebene Vektorlängen zurück. Der Benchmark ist stark von der Architektur des Speichersubsystems abhängig. Arbeitet der Cache beispielsweise mit der write–back–Strategie, erhöht sich die gemessene Speicherperformance deutlich. Ein write–back–Cache speichert alle Daten, die neu geschrieben werden, erst im Cache, und schreibt sie wieder zurück in den Hauptspeicher, sobald die Speicherzelle im Cache durch andere Daten belegt werden soll. Im Gegensatz dazu schreibt ein write–through–Cache die gelieferten Daten immer sofort in den Hauptspeicher [SITE-2]. Außerdem sind in einige Systeme noch spezielle Puffer integriert, welche ein mehrfaches Schreiben zwischenspeichern. Der Pseudocode für diesen Test hat folgende Form:

```
for all vector length // wiederhole folgende Anweisungen für jede Vektorlänge
  timer start
  for iteration count // wiederhole für die vorgegebene Anzahl folgende Anweisungen
    for I = 0 to vector length // für alle Datenblöcke bis zur aktuellen Vektorlänge
      memory[I] = register // schreibe einen Datenblock in den Hauptspeicher
```

```
  timer stop
```

2.3.3.4 Cache Read/Modify/Write

Dieser Benchmark errechnet die Geschwindigkeit, mit der Daten verschiedener Vektorlängen gelesen, verändert und zurückgeschrieben werden können. Bei diesem Test wird gegenüber Cache Read und Cache Write der doppelte Datenverkehr erzeugt, da die Daten vom Speicher/Cache gelesen und anschließend in den Speicher/Cache geschrieben werden müssen. Jede Richtung des Datentransports wird für die Berechnung der Speicherbandbreite mit einbezogen. Die ermittelte Speicherbandbreite dieses Tests ist höher gelegen als Cache Read oder Write, da der Compiler die Operationen besser einteilen und die Speicherzugriffe gruppieren kann, um die Kosten für diese zu minimieren. Der Pseudocode für Cache Read/Modify/Write lautet:

```
for all vector length // wiederhole folgende Anweisungen für jede Vektorlänge
  timer start
  for iteration count // wiederhole für die vorgegebene Anzahl folgende Anweisungen
    for I = 0 to vector length // für alle Datenblöcke bis zur aktuellen Vektorlänge
      memory[I]++ // lies einen Datenblock aus dem Speicher , erhöhe
                  // seinen Wert um eins und schreibe ihn zurück in den Speicher
  timer stop
```

2.3.3.5 handtuned - Versionen

Die Tests Cache Read, Cache Write und Cache Read/Modify/Write sind als handtuned–Tests ausführbar. In diesen Versionen der Tests fließen einige Optimierungen ein:

- achtfaches Entrollen der Schleifen: Das Entrollen von Schleifen soll eine Erhöhung der Geschwindigkeit zur Folge haben, da mehr gleichartige Operationen pro Schleifendurchlauf ausgeführt und damit dem Compiler mehr Ansätze für Optimierungen geboten werden [SITE-3]. Mit jedem Schleifendurchlauf wird auf acht Datenblöcke zugegriffen.

- Jede Operation ist unabhängig von den vorherigen sieben Operationen. Dadurch kann der Code auf Prozessoren mit Pipelines effizienter ausgeführt werden.

- Wiederverwendung von Registern: Die Register sind zu bestimmten Blöcken im Speicher fest zugewiesen. Diese Blöcke werden so oft wie möglich für Operationen genutzt.

Die mit diesen Optimierungen erreichten Werte spiegeln das Maß wieder, das auch mit Hilfe eines Compilers, der diese Arten der Optimierungen unterstützt, an diesen Schleifen minimal erreicht werden sollte. Die Komplexität der Schleifen der nicht handtuned Versionen ist niedrig, somit sollte[22] der Compiler sie entrollen und optimieren können.

[22] je nach Fähigkeiten des Compilers

2.3.3.6 Memory Set

Die C-Bibliothek beinhaltet die Funktion *memset()*, um Bereiche des Speichers initialisieren zu können. Da sie oft von Betriebssystem und Programmen genutzt wird, ist die Funktion hoch optimiert. Die Optimierung äußert sich darin, dass die Funktion während der Übersetzung durch Assemblercode ersetzt wird (dies ist zum Beispiel beim Compiler gcc der Fall). Manche Rechnersysteme beinhalten zusätzliche Hardware, um diese Operation auszuführen zu können, speziell wenn der Wert, welcher gesetzt werden muss, null ist. Das durch diesen Test ermittelte Ergebnis kann mit den Werten, welche die beiden Cache Write Tests zurückliefern, verglichen werden. Auf einigen Systemen wird bei beiden Cache Write Versionen auf Grund von Compileroptimierungen die *memset()* Funktion aufgerufen. Der Pseudocode dieses Tests sieht folgendermaßen aus:

```
for all vector length // wiederhole folgende Anweisungen für jede Vektorlänge
  timer start
  for iteration count // wiederhole für die vorgegebene Anzahl folgende Anweisungen
     memset(vector1,0xf0,length) // schreibe die Anzahl von length Datenblöcken mit
          dem Hexadezimalwert f in den Hauptspeicher
  timer stop
```

2.3.3.7 Memory Copy

memcpy() ist eine weitere Funktion der C-Bibliothek, welche häufig genutzt wird. Sie kopiert Bereiche des Speichers. Diese Funktion wird ebenfalls meist zur Übersetzungszeit durch Assemblercode ersetzt. Das Kopieren von Speicherbereichen erfordert zwei Operationen: das Lesen und Schreiben vom und in den Speicher. Somit sind die gelieferten Ergebnisse mit denen der beiden Cache Read/Modify/Write–Tests vergleichbar. Der Pseudocode dieses Tests hat folgende Struktur:

```
for all vector length // wiederhole folgende Anweisungen für jede Vektorlänge
  timer start
  for iteration count // wiederhole für die vorgegebene Anzahl folgende Anweisungen
     memcpy(dest,src,vector length) // kopiere die Anzahl von vector length Bytes
          von src nach dest
  timer stop
```

2.3.3.8 Übersetzung von cachebench

cachebench wird durch das im Quellpaket enthaltene Makefile mit den Compileroptionen “-O2 -Wall -DREGISTER” übersetzt. Die Option “-DREGISTER” drückt den Versuch aus, bestimmte Variablen in Registern des Prozessors zu halten. Damit der Compiler den Nutzer über Mängel und Unstimmigkeiten im Quelltext warnt, wird die Option “-Wall” übergeben. Weiterhin besitzt der Compiler verschiedene Optimierungsstufen. Mit “-O2” wird eine solche Optimierungsstufe gewählt. Je höher die Zahl der gewählten Optimierungsstufe, umso mehr Optimierungsprozeduren werden am Code vollzogen. Eine Aufzählung der Optimierungsstufen und -prozeduren würde an dieser Stelle zu weit führen [SITE-4]. Standardmäßig wird **cachebench** so compiliert, dass es den Datentyp Double verwendet. Über folgende Compileroptionen kann die Verwendung eines Datentyps erzwungen werden:

-DUSE_INT: Datentyp Integer

-DUSE_FLOAT: Datentyp Float

-DUSE_CHAR: Datentyp Character

-DUSE_DOUBLE: Datentyp Double

2.3.3.9 Kommandozeilenargumente für cachebench

r: Cache Read

w: Cache Write

b: Cache Read/Modify/Write

t: Nutzung der handtuned Versionen (diese Option in Verbindung mit “r”, “w” oder “b”)

s: memset() Benchmark

p: memcpy() Benchmark

m: Angabe des Zweierlogarithmus der maximalen Länge der zu testenden Vektoren (beispielsweise 23 für 8 MByte)

x: Angabe der Anzahl von Messungen, welche zwischen zwei Zweierpotenzen vorzunehmen sind

d: Angabe der Sekunden pro Messung

e: Angabe der Anzahl von Wiederholungen jeder Messung (für die jeweils vorgegebene Dauer)

2.3.3.10 Begründung der Aufnahme in die Qualip-Suite

cachebench bietet sich für die Überprüfung der Geschwindigkeit des Arbeitsspeichers an. Außerdem lässt sich die Größe der Cache–Speicher praktisch ermitteln. Jedoch ist dies nicht bei jedem Rechner möglich, da anhand der gelieferten Werte keine Stufe bei Überschreitung der Grenzen der Cache–Ebenen zu erkennen ist oder auf Grund eines Abfallens des Datendurchsatzes bei Datensatzgrößen, die nicht das Ende einer Cache–Ebene darstellen (siehe dazu das nachfolgende Unterkapitel 2.3.3.11). Die Ergebnisse des Benchmarks sind reproduzierbar. **cachebench** ist der einzige Benchmark, welcher unter Linux den Speicherdurchsatz bei ansteigender Datensatzgröße ermittelt. Mit dem Programm **stream** (Kapitel 2.3.1) ist es ebenfalls möglich, den Datendurchsatz mit Testgrößen auszumessen, die kleiner als 1st–Level–Cache sind. Jedoch arbeitet das Programm die Schleifen dann schneller ab als der Autor des Programms auf Grund von erhöhten Messungenauigkeiten empfiehlt. **stream** führt seine Messungen für eine durch das Programm vorgegebene Zahl von Wiederholungen aus, **cachebench** dagegen für eine durch den Nutzer gewählte Zeit.

2.3.3.11 Testergebnisse und Erläuterungen

Testrechner

Die Konfiguration der in den Tests verwendeten Rechner ist in Tabelle 2.4 einzusehen.

Rechnername	mailand	hornisse	dallas	dakar
Prozessor	Intel Pent.3	Intel Pent.4	Intel Pent.4	AMD Athlon XP2100+
Taktfrequenz [MHz]	867	1600	2400	1733
1st-Level-Cache[23][KByte]	32	8	8	64
2nd-Level-Cache[24][KByte]	256	512	512	256
Frontside-Bus [MHz][25]	133	400	533	266
RAM-Typ	SDR	DDR	RDRAM	DDR
RAM-Taktfrequenz [MHz][25]	133	266	400	333

Tabelle 2.4: Komponenten der zu Tests von **cachebench** verwendeten Rechner

Tests mit variierenden Compileroptionen

Die Möglichkeiten der Tests mit **cachebench** sind sehr vielfältig: Es existieren acht Tests, welche jeweils mit vier verschiedenen Datentypen durchgeführt werden können. Desweiteren haben die Compileroptionen einen starken Einfluss wie in Abbildung 2.1 erkennbar ist. Die Veränderungen, die eine neuere Version des Compilers besonders in Zusammenspiel mit zukünftiger Hardware verursacht, sind schwer abzuschätzen. Ziel war es demnach, möglichst wenige Optimierungen des Compilers zu verwenden und somit den Einfluss des Compilers gering zu halten.

Die Werte in Abbildung 2.1 wurden mit dem Rechner *mailand* (siehe Tabelle 2.4) erzielt. Die genutzten Kommandozeilen–Optionen des Programms **cachebench** waren für die handoptimierten–Tests
<“-m 24 -e 1 -x 4 -d 5 -tb”>, für die anderen Tests <“-m 24 -e 1 -x 4 -d 5 -b”>. Die handoptimierten Tests sind in die Abbildung eingeflossen, da hier schon innerhalb des Quelltextes von **cachebench** Optimierungen vorgenommen wurden, die den Optimierungen des Compilers entsprechen.

In Abbildung 2.1 ist deutlich zu erkennen, dass bei den beiden Tests mit niedrigerer Stufe der Optimierung (<“-DUSE_INT”> und <“-O2 -DREGISTER -DUSE_INT”>) die handoptimierten Varianten einen wesentlich erhöhten Datendurchsatz erwirken. Auch ist die Stufe zwischen 1st- und 2nd-Level-Cache bei 16 KByte Testgröße nur bei drei der sechs Tests zu erkennen. Von diesen drei Tests ist wiederum nur bei zwei Tests die Grenze zwischen 2nd-Level-Cache und Hauptspeicher bei 256 KByte Testgröße an der annähernd richtigen Stelle verzeichnet. Von den beiden verbleibenden Kurven ist eine durch den

[23] nur der Datenteil des 1st-Level-Cache

[24] 2nd-Level-Cache ist bei allen diesen Prozessoren auf dem Prozessor-Die und wird mit vollem Prozessortakt angesprochen

[25] Wenn pro Pin und Takt mehr als ein Bit Daten übertragen werden, so wird statt der tatsächlichen die um diesen Faktor erhöhte Frequenz angegeben.

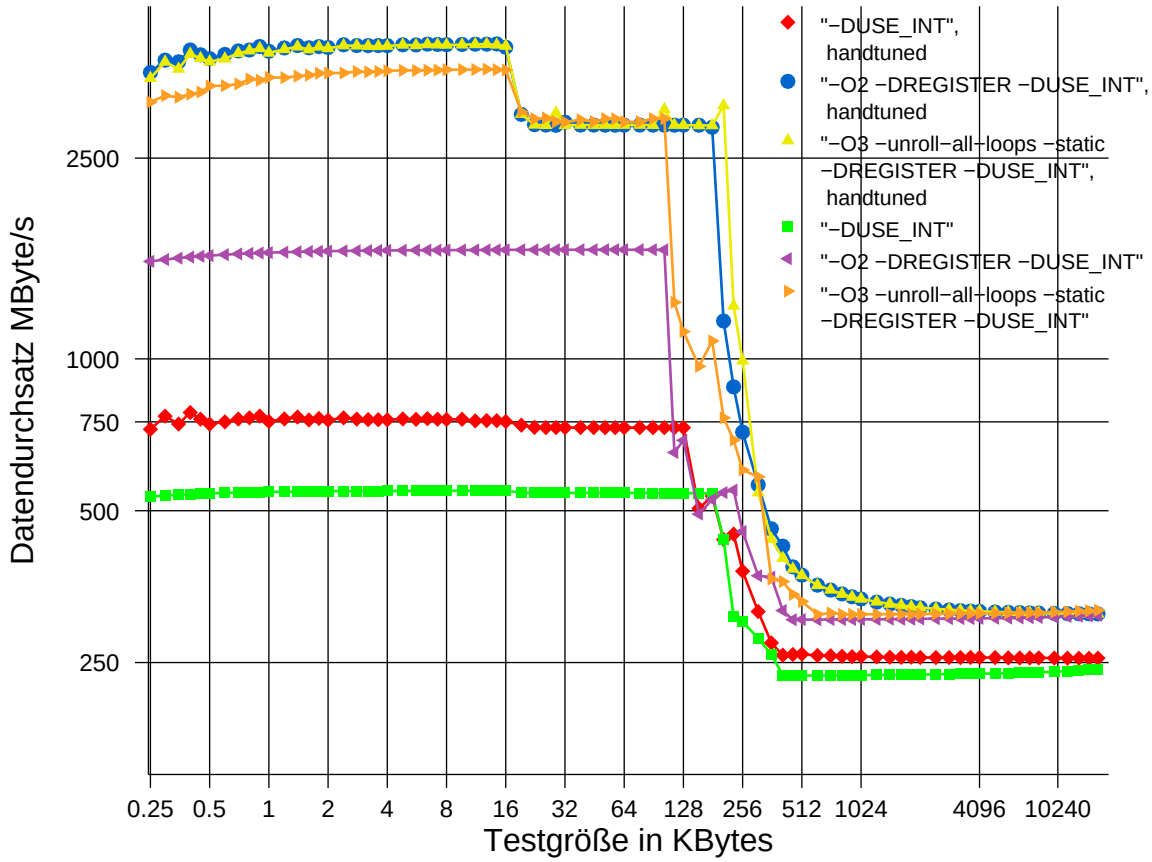


Abbildung 2.1: **cachebench**, Read/Modify/Write-Test bei Nutzung des Datentyps Int auf Rechner *mailand* mit unterschiedlichen Compileroptionen

Compiler höher optimiert. Da es ein weiteres Ziel ist, den Compiler so wenig wie möglich Optimierungsversuche unternehmen zu lassen, wird für diesen Rechner der handoptimierte Test mit den Compileroptionen <“-O2 -DREGISTER -DUSE_INT”> festgelegt.

Bei der Wahl des geeigneten Tests für die Suite war ein Treppen–förmiger Verlauf der Kurve, welche sich aus den Ergebnis-Werten eines **cachebench**–Laufs ergibt, von Bedeutung. Bei einem Treppen–förmigen Verlauf ist die Kurve immer parallel zur X- bzw. Y-Achse. Jede Stufe beschreibt dabei die Eigenschaften Größe und Geschwindigkeit je einer Cache-Ebene bzw. des Hauptspeichers des Rechnersystems.

Mögliche Ursachen für den verringerten Datendurchsatz bei niedriger Optimierungsstufe des Compilers ist eine ungeeignete Strukturierung der Anweisungen, so dass die Pipeline–Stufen der CPU nicht genutzt werden. Weiterhin kann die nicht vorhandene Stufenform der Kurve auf eine nicht ideale Nutzung der Register bzw. Cachespeicher zurückzuführen sein. Der Compiler muss entscheiden, welche Variablen in Registern gehalten werden. Müssen Werte von Variablen, welche im Vergleich zu den anderen Variablen häufig benötigt werden, aus dem 2nd–Level–Cache oder gar dem Hauptspeicher geladen werden, so muss die CPU Wartezyklen ausführen. Der Datendurchsatz sinkt auf das Niveau des 2nd–Level–Cache bzw. Hauptspeichers ab.

Einfluss der verschiedenen Datentypen

In Abbildung 2.2 wird der Einfluss der vier verschiedenen Datentypen, mit denen **cachebench** die Tests durchführt, verdeutlicht. Abbildung 2.2 zeigt den Rechner *dallas* (siehe Tabelle 2.4).

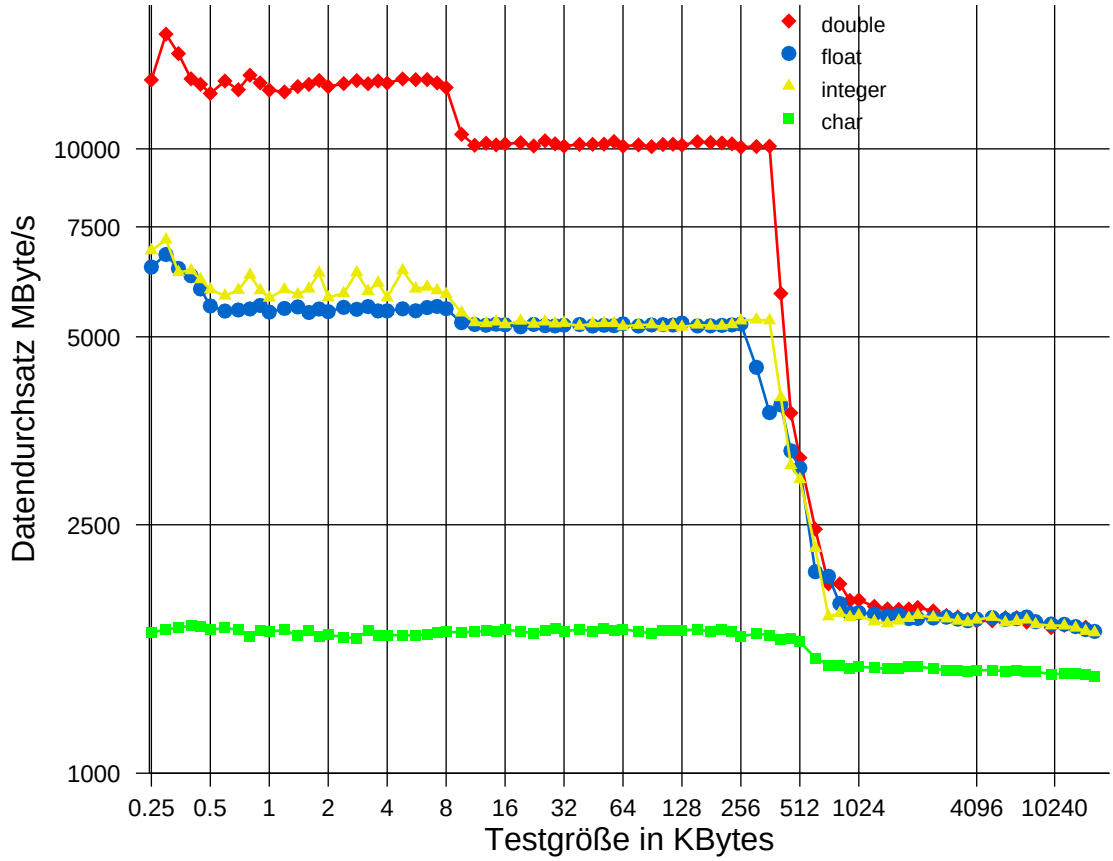


Abbildung 2.2: **cachebench**, handtuned Read/Modify/Write-Test auf Rechner *dallas* bei variierendem Datentyp

cachebench wurde mit den Compileroptionen
<“-O2 -DREGISTER -DUSE_XXXX”> übersetzt und mit den Optionen
<“-m 24 -e 1 -x 4 -d 5 -tb”> ausgeführt. Dabei wurde “XXXX” durch den jeweiligen Datentyp ersetzt. Für diesen Rechner ist der Datentyp Double am besten geeignet. Wie aus der Abbildung 2.2 ersichtlich, ist hier der Übergang vom 1st- zum 2nd-Level–Cache bei einer Testgröße von acht KByte am deutlichsten ausgeprägt. Allerdings hat die Verwendung dieses Datentyps den Nachteil, dass dieser 64-Bit[26] groß ist. Die x86-CPUs können nur arithmetische Operationen mit 32-Bit[27] ausführen. Es ergibt sich somit ein höheres Risiko, mit diesem Datentyp zu arbeiten, da unbekannt ist, zu welchen Testergebnissen (insbesondere, ob diese noch eine Treppenform nachbilden) er auf noch nicht getesteten Rechnertypen führt. Der Datentyp Char liefert die am wenigsten stufenförmige Kurve. Dieser Datentyp ist auch bei anderen, hier nicht vorgestellten Rechnern ähnlich ungeeignet zur Ermittlung der Cachegrößen und -geschwindigkeiten.

[26] zumindest bei den getesteten Rechnern und dem Compiler **gcc**, Version 2.95

[27] Ausnahmen sind die Operationen, welche durch die MMX- bzw. SSE2–Einheiten ausgeführt werden. Diese wurden jedoch nicht bei der Ausführung von cachebench genutzt.

Testergebnisse unterschiedlicher Hardware

Auf Rechnersystemen mit unterschiedlicher Architektur stellten sich jeweils andere Tests des Programms **cachebench** am geeignetsten dar. Die Daten der getesteten Rechner können in Tabelle 2.4 eingesehen werden.

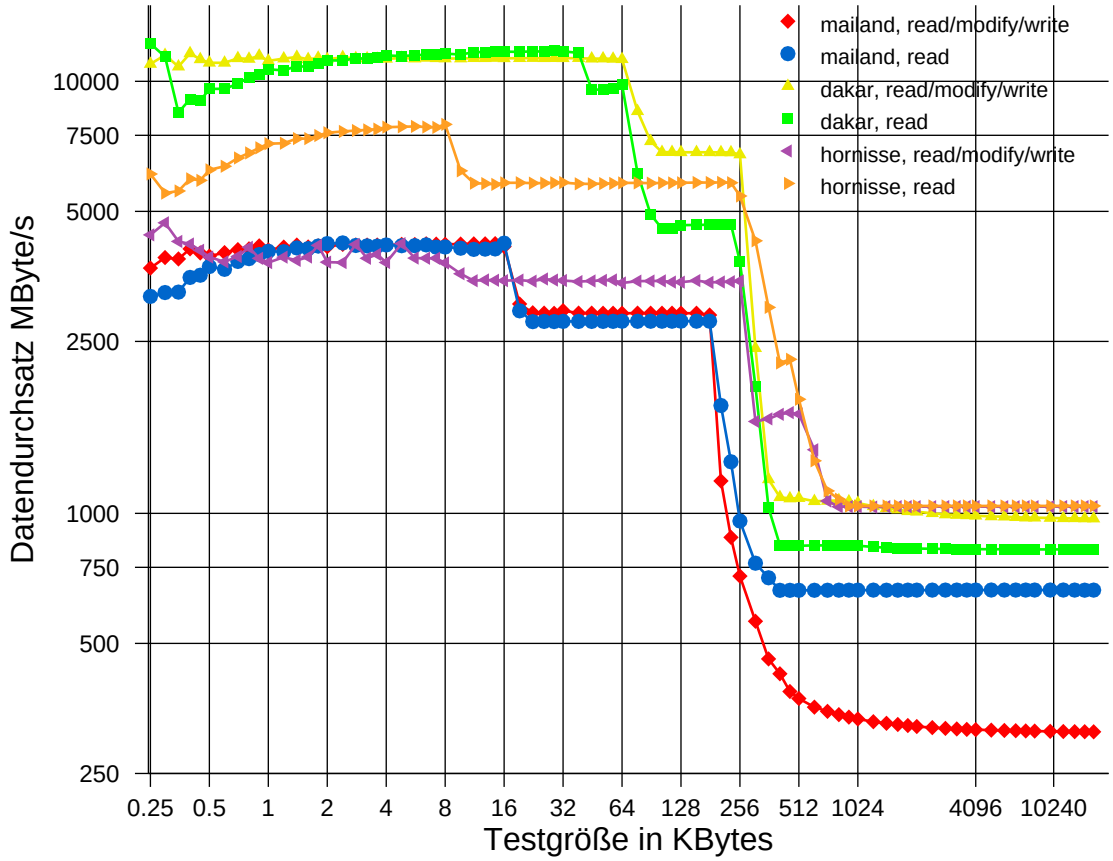


Abbildung 2.3: **cachebench**, handtuned-Versionen des Read/Modify/Write- und Read-Tests unter Nutzung des Datentyps Int auf unterschiedlichen Rechnersystemen

Die für die ermittelten Testergebnisse genutzten Compileroptionen sind <“-O2 -DREGISTER -DUSE_INT”>, die **cachebench**–Optionen <“-m 24 -e 1 -x 4 -d 5 -tb”> bzw. <“-m 24 -e 1 -x 4 -d 5 -tr”>, es handelt sich somit immer um handoptimierte Versionen der Tests. Die Cache-Architektur auf dem Rechner *hornisse* ist nur mit dem Read-Test, nicht mit dem Read/Modify/Write-Test erkennbar. Auf dem Rechner *dakar* ist dagegen bei dem Read-Test innerhalb des Bereiches des 1st-Level-Caches eine weitere Stufe zwischen 46 und 64 KByte Testgröße verzeichnet. Somit stellt der Read/Modify/Write–Test die Cache-Architektur dieses Rechners besser dar als der Read–Test. Ein Programm, welches auf Basis der Ergebnisse von **cachebench** die Größen der einzelnen Caches ermitteln soll, könnte bei dem Read–Test auf dem Rechner *dakar* das Ende des 1st–Level–Caches schon bei 46 KByte statt bei 64 KByte ermitteln.

Vergleiche der Ergebnisse mit anderen Benchmarks

Bei der Untersuchung der Geschwindigkeit von Hauptspeicher und Cache, zeigte sich ein sehr differenziertes Bild der Ergebnisse von unterschiedlichen Programmen auf dem gleichen Testrechner. Die Tests wurden mit den Programmen **stream**, **memtest86** (eigenes Betriebssystem), **cachemem** und **memspd** (beide unter DOS) durchgeführt.

Für die Werte, welche in Abbildung 2.4 dargestellt wurden, gelten folgende Bedingungen. Mit dem Programm **cachebench** wurde die handoptimierte Version des Read/Modify/Write–Benchmarks genutzt. **cachebench** selber wurde mit **gcc**, Version 2.95 und den Optionen < -O2 -DREGISTER -DUSE_INT> übersetzt. Somit wurde vorgegeben, weitestgehend die Register der CPU zu nutzen und der Datentyp Integer verwendet. Die Werte des Programms **stream** sind Durchschnitte der vier Ergebnisse. Das Programm **cachemem** lieferte Ergebnisse für den Lese- und Schreibmodus. Es wurden die Durchschnittswerte von Lese- und Schreibmodus im Diagramm verwendet. Als Testsystem kam der Rechner *dakar* (siehe Tabelle 2.4) zum Einsatz.

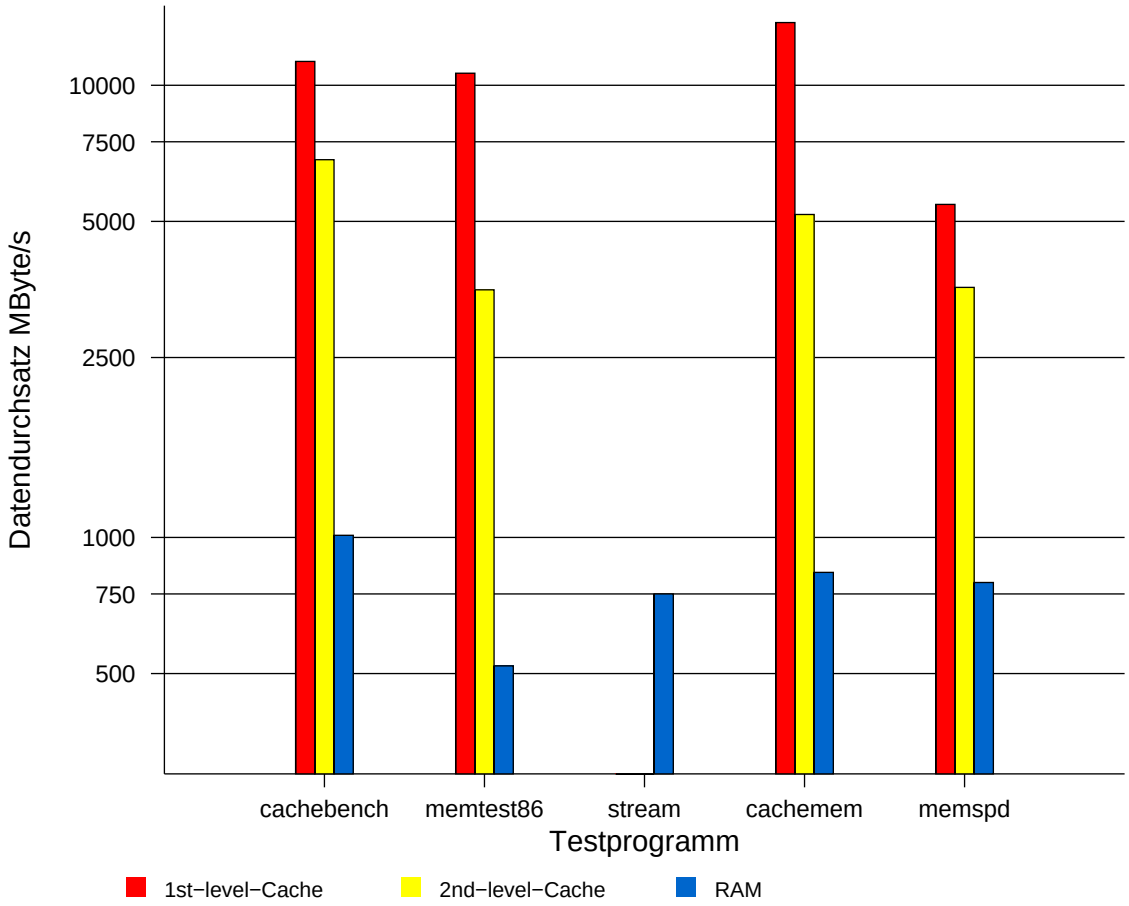


Abbildung 2.4: Cache- und Hauptspeicher-Benchmarks im Vergleich auf Rechner *dakar*

Die genaue Bezeichnung des im Testrechner verbauten Speicher lautet PC2700-2533. Er ist mit 166 MHz getaktet und hat eine CAS–Latency[28] von 2.5 Takten, was 15 ns entspricht. Die maximale theoretische Übertragungsgeschwindigkeit mit diesen Speichermodulen wurde vom Hersteller mit 2.67 GByte/s angegeben. Dieser Wert entspricht nicht den

[28] siehe Kapitel 1.3.3

im praktischen Einsatz erzielten Transferraten [c't_08/2002]. Jedes Programm errechnet, je nachdem zu welchem Zeitpunkt auf welche Daten zugegriffen wird, unterschiedliche Datentransferraten. Eine theoretische Berechnung der Transferrate von RAM–Bausteinen mit der Spezifikation PC2100-2022 führte in [c't_08/2002] zu einer Transferrate von 776 MByte/s. Der Hersteller hat für diesen Chip eine Übertragungsrate von 2,13 GByte/s angegeben. Die unterschiedlichen Datentransferraten erklären sich durch Verteilung der Daten im Speicher und durch die Wartezeit, zwischen der Anforderung und dem Anliegen am Bus der ersten Daten. Der Programmierer (bzw. die CPU bei AMD Athlon XP und Intel Pentium 4) kann diese Wartezeiten kompensieren, indem er die Prefetching–Fähigkeiten der CPU's ausnutzt. Dies erfolgt, indem die Daten vor Gebrauch, während von der CPU andere Berechnungen ausführt, in den CPU-Cache kopiert werden.

Die Abbildung 2.5 veranschaulicht die Auswirkungen, welche Veränderungen im BIOS des Rechners bewirken. Grundlage dieser Ergebnisse ist der Rechner *mailand*, wobei **cachebench** mit den Parametern <"-O2 -DREGISTER -DUSE_DOUBLE"> übersetzt und <"-m 24 -e 1 -x 4 -d 5 -tr"> ausgeführt wurde.

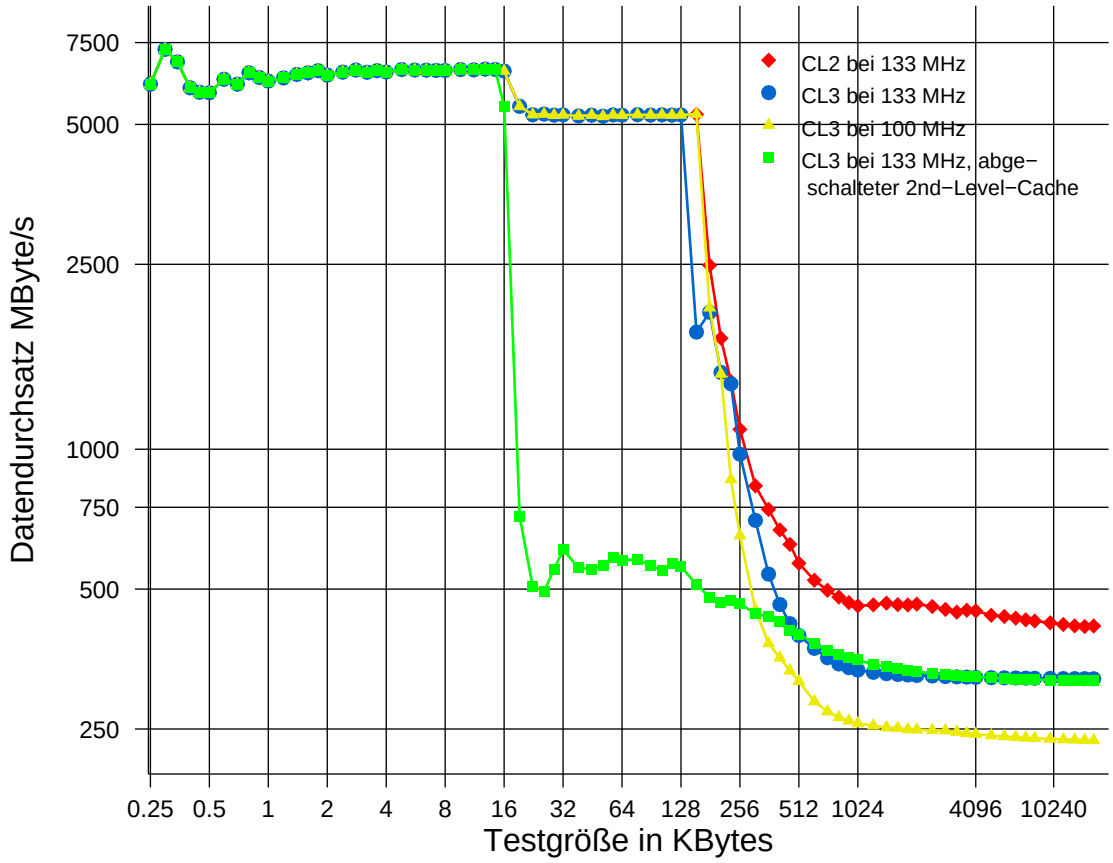


Abbildung 2.5: **cachebench**, handtuned Read-Test und Datentyp Double auf Rechner *mailand* bei variierenden BIOS-Parametern

Im Rechner *mailand* sind 2 SDR–SDRAM–Module der Firma Infineon eingebaut, welche die Spezifikation PC133-222[29] besitzen. Es wurden, wie in Abbildung 2.5 zu se-

[29] 133 MHz, jeweils 2 Takte (entspricht 15 ns) CAS–Latency, RAS- to CAS–Delay und RAS Precharge Time (siehe Kapitel 1.3.3)

hen, folgende Veränderungen im BIOS durchgeführt:

- Erhöhen der CAS–Latency[30] auf 3
- Erhöhen der CAS–Latency auf 3 und Herabsetzen der Taktfrequenz des Speichers auf 100 MHz
- Erhöhen der CAS–Latency auf 3 und Abschalten des 2nd–Level–Caches

In der Abbildung ist zu erkennen, dass eine Verringerung der Taktfrequenz des Speichers bzw. eine Erhöhung der CAS–Latency eine Verringerung des Datendurchsatzes bewirken. Bei deaktivierten 2nd-Level-Cache ist die Stufe der Kurve im Diagramm zwischen 16 und 160 KByte nicht mehr vorhanden.

2.4 Die Grafikkarte

2.4.1 SPEC glperf

Dieses Programm wurde zur Messung der Ausführungsgeschwindigkeit von zwei- und dreidimensionalen OpenGL–Befehlen geschaffen. Diese Befehle dienen der Darstellung von Primitiven wie Punkte, Linien oder Dreiecke. Es dient nicht der Darstellung von Modellen. Das Programm kann, vergleichbar mit **x11perf**, für die Evaluierung von Treibern für Grafikkarten eingesetzt werden. **SPEC glperf** kann auch zur Ausgabe von Ergebnissen (Graphen) auf Basis von OpenGL genutzt werden. Es wurde eine Sprache geschaffen, unter deren Nutzung Darstellungen von Primitiven programmierbar sind. Neben dem Programm selbst bietet die SPEC–Organisation auch so genannte Skripte an. Dies sind Quelltexte dieser Sprache, welche als Beispiele dienen. Diese 13 Skript–Dateien messen jeweils ein spezielles Merkmal von OpenGL. Eine Zahl geeigneter[31] Darstellungen zu programmieren, ist auf Grund des Zeitaufwands im Rahmen dieser Ausarbeitung nicht möglich. Demnach wird **SPEC glperf** in der Qualip–Suite keine Anwendung finden.

2.4.2 SPEC viewperf

SPEC viewperf ist ein Programm zur Messung der Geschwindigkeit der Darstellung von OpenGL–basierten Objekten. Es analysiert Daten und wandelt diese in OpenGL–Befehle um. Diese werden an das X-Windows System weitergegeben, welche diese Befehle mit Hilfe der OpenGL–Bibliotheken anzeigen kann. Die Daten werden für eine spezifizierte Anzahl darzustellender Bilder oder Zeit gerendert. **SPEC viewperf** liefert die Anzahl der erreichten Bilder pro Sekunde und den genutzten OpenGL - Renderer auf dem Standard–Ausgabestrom zurück.

Zu dem **SPEC viewperf**–Programm bietet die SPEC–Organisation sog. Viewsets an. Diese beinhalten Angaben zum darzustellenden Modell, Texturen und Bewegungsabläufen. Ein Viewset enthält mehrere Tests, die sich in Modell, Textur oder Bewegungs-

[30] das BIOS des Rechners erlaubte keine Einstellung des RAS- to CAS–Delay und der RAS Precharge Time

[31] Testen einer Auswahl der Möglichkeiten von OpenGL, Tests für in der Praxis häufiger angewendete Funktionen mit höherer Wertigkeit einfließen lassen

ablauf unterscheiden. Jeder Test ist mit einer Wichtung versehen worden. Aus den jeweiligen Ergebnis der einzelnen Tests *x* (Bilder pro Sekunde) und der Wichtung *y* errechnet sich dann das Ergebnis für das komplette Viewset *g*, welches aus *z* Tests besteht.

$$g = \prod_{i=0}^{z} x^{y}$$

Die durch die Viewsets erzeugten OpenGL–Befehle entstammen Programmen wie **Advanced Visualizer** der Firma Alias/Wavefront, sowie **Visualization Data Explorer** der Firma IBM. **SPEC viewperf** ist somit der Gruppe der Anwendungsbenchmarks zuzuordnen. Die Viewsets wurden nicht von der SPEC–Organisation entwickelt, sondern von den Firmen der Programme, welche das Viewset für Performance–Messungen einsetzen.

2.4.2.1 Vergleich Version 6.1.2 vs 7.0

SPEC viewperf Version 7.0 des Programms wird als komplettes Paket mit sechs Viewsets geliefert. Das Software–Paket benötigt dekomprimiert 809 MByte. Die Version 7.0 wirkt unausgereift. In der Datei "*makefile.linux*" (zum Kompilieren des **viewperf**–Programms selber) ist eine Datei fehlerhaft benannt. Erst nach Umbenennen der Datei oder Änderung des Makefiles lässt sich **SPEC viewperf** übersetzen. Beim Übersetzen der beiden Versionen meldet der Compiler Warnungen. Bei Version 6.1.2 beschränken sich diese Warnungen auf 2 Header–Dateien, welche mittels include in die C-Quelltexte eingebunden wurden. Diese Dateien enthalten mehr öffnende geschweifte Klammern als geschlossene. Die Headerdateien sind somit unvollständig. Auch in Version 7.0 sind diese Header–Dateien nicht vervollständigt worden. Jedoch liefert der Compiler bei dieser Version zusätzlich vier weitere Arten von Warnungs–Meldungen zurück. Trotz der Warnungen des Compilers sind während der Testläufe keine Abstürze der Rechnersysteme aufgetreten, welche auf das Programm zurückzuführen waren. Aus genannten Gründen ist die Version 6.1.2 der Version 7.0 vorzuziehen.

2.4.2.2 Wahl eines Viewsets

Es wurde das Viewset "MedMCAD" (siehe Abbildung 2.6) für diesen Test genutzt. Dieses Viewset nutzt für die Darstellung 349000 Dreiecke, was der höchste Wert aller Viewsets ist. Es repräsentiert im Gegensatz zu den anderen Viewsets nicht nur ein einzelnes Grafik–Programm, sondern eine Gruppe von CAD–Programmen wie Pro/Engineer der Firma PTC und SolidWorks der Firma SolidWorks Corporation. Es werden zwölf Tests ausgeführt, wobei das Modell eines Kopierers als Drahtmodell oder mit schattierter Oberfläche dargestellt wird. Die Tests unterscheiden sich in Perspektive, Detailreichtum, Lichtquellen und Nutzung einer Clipplane[32].

[32]Die Clipplane ist eine Ebene im Raum. Haben das Objekt und die Clipplane gemeinsame Punkte, so wird das Objekt nur bis zur Clipplane dargestellt (es wird "abgeschnitten").

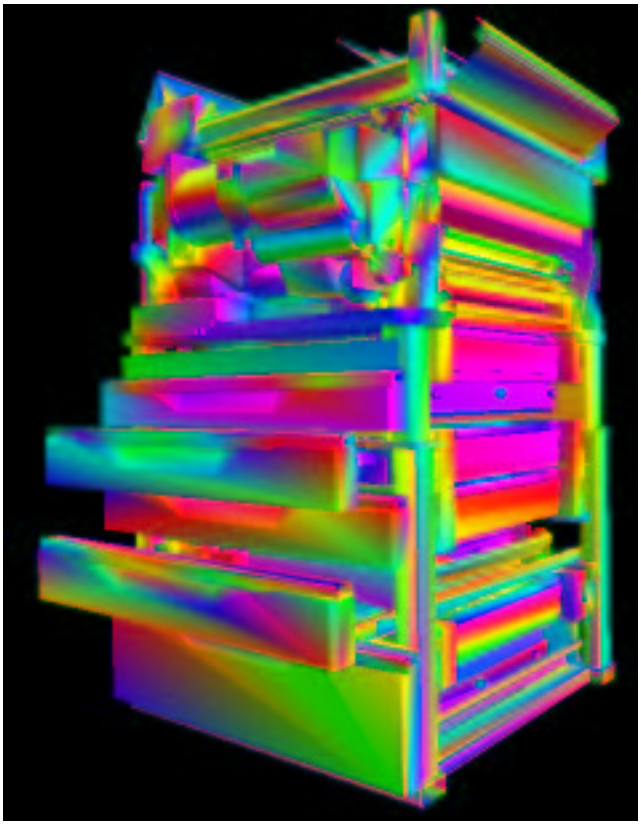


Abbildung 2.6: Test 10 des Programms SPEC **viewperf**

2.4.2.3 Testergebnisse

In Tabelle 2.5 sind die Rechner, in Tabelle 2.6 die Grafikkarten verzeichnet, die bei den Tests verwendet wurden. Tabelle 2.7 enthält die Ergebnisse der **viewperf**-Durchläufe.

Rechner	1	2	3
CPU [MHz]	AMD Duron 750	AMD Athlon XP 1300	Intel Pentium 4 2400
Frontside-Bus [MHz][33]	200	200	533
Hauptspeicher [MByte]	PC133 768	PC2700 768	PC2700 768

Tabelle 2.5: Testrechner für **viewperf**-Tests

Bei den Tests, welche die Clipplane-Funktion nutzen, fällt die Bildwiederholrate auf das Niveau des Software–Render–Modus (deaktivierte Hardware–Beschleunigung) zurück. Die Nutzung der Clipplane wird deshalb im Skript des Viewsets deaktiviert. Mit dem **viewperf**–Benchmark soll ermittelt werden, ob OpenGL–Befehle unter Nutzung der Hardware–Beschleunigung ausgeführt werden. Welche OpenGL–Befehle die Grafikkarte unterstützt, ist für die Qualip–Suite nicht relevant.

[33] Wenn pro Pin und Takt mehr als ein Bit Daten übertragen werden, so wird statt der tatsächlichen die um diesen Faktor erhöhte Frequenz angegeben.

Grafikkarte	1	2
Hersteller & Modell	Elsa Gladiac MX	Asus V8170 DDR
Grafikkartenchip	Nvidia Geforce 2MX400	Nvidia Geforce 4MX440

Tabelle 2.6: Grafikkarten für **viewperf**-Tests

Die zur Darstellung von Bildern notwendigen Berechnungen können als Pipeline (auch Rendering Pipeline genannt) aufgefasst werden. Die Berechnungen in den einzelnen Stufen der Rendering Pipeline werden von CPU und Grafikkarte durchgeführt. Jede Stufe vollzieht eine Funktion, die zum Rendern des Bildes notwendig ist. Dabei bestimmt diejenige Stufe der Pipeline, welche die meiste Zeit zur Ausführung ihrer Funktion benötigt, die Geschwindigkeit (den Takt der dargestellten Bilder pro Sekunde) [MölHai]. Befindet sich die langsamste Stufe der Rendering Pipeline innerhalb der Grafikkarte, so wird das selbe Rechnersystem mit höher getakteter CPU keine höhere Zahl an Bildern pro Sekunde darstellen. In den Ergebnissen (denen ohne Clipplane) ist zu erkennen, dass mit der Grafikkarte 1 die höchste nur etwa 15 % über der niedrigsten erreichten Punktzahl liegt. Ist die langsamste Stufe die CPU, so wird sich mit Erhöhung der Leistungsfähigkeit der CPU bis zu einem bestimmten Grenzwert auch die Anzahl der Bilder pro Sekunde erhöhen. Bei Rechner 1 beträgt der Abstand zwischen den mit beiden Grafikkarten erreichten Punktzahlen (ohne Clipplane) 1.8 Punkte. Bei Rechner 3 sind es 17.6 Punkte. Die langsamste Stufe der Rendering Pipeline ist bei Rechner 1 in Kombination mit Grafikkarte 2 somit die CPU.

Rechner	1	1	2	2	3	3	3
Grafikkarte	1	2	1	2	1	2	1
Grafik-Treiber[34]	N	N	N	N	N	N	M
Test 1 [fps]	59.4	65.3	59.3	87.5	59.2	89.8	5.9
Test 2 [fps]	60.1	65.4	61.0	88.9	61.1	94.9	5.7
Test 3 [fps][35]	43.8/12.0	48.1/12.0	45.6/19.6	62.6/19.6	45.7/27.9	80.4/-	4.5/6.5
Test 4 [fps]	43.5	48.1	45.8	62.8	46.4	82.6	4.3
Test 5 [fps]	18.2	19.4	20.8	27.7	20.9	34.8	1.6
Test 6 [fps][35]	16.8/1.4	18.0/1.4	20.6/2.4	27.6/2.4	21.1/3.2	35.0/-	1.5/2.6
Test 7 [fps]	14.8	15.1	18.6	21.6	19.3	32.4	1.5
Test 8 [fps]	14.3	15.4	14.5	21.5	14.5	23.9	1.9
Test 9 [fps]	12.6	13.2	17.4	19.3	19.3	26.3	1.4
Test 10 [fps][35]	12.6/1.1	13.2/1.1	16.9/1.8	19.3/1.8	19.4/2.5	26.2/-	1.2/2.2
Test 11 [fps]	17.1	18.3	18.0	27.6	18.0	31.4	1.5
Test 12 [fps]	17.3	18.0	20.8	27.7	20.8	35.8	1.9
Punktzahl[35]	24.8/15.0	26.6/15.8	27.9/18.4	37.5/23.3	28.6/20.3	46.2/-	2.5/2.8

Tabelle 2.7: Testergebnisse des Viewsets MedCAD-01 des Programms **viewperf** bei unterschiedlicher Hardware und variierendem Testablauf

2.4.2.4 Begründung für die Aufnahme in die Qualip-Suite

Das Programm **SPEC viewperf** ist zur Entstehungszeit dieser Arbeit der einzige verfügbare OpenGL–Benchmark, welcher Szenarien mitliefert. Das Programmpaket hinterlässt im Hinblick auf die Warnungen des Compilers einen unausgereiften Eindruck. Da es dennoch während der Ausführung keine Instabilitäten zeigt, fließt es in die Qualip-Suite ein. Durch die Wahl von bestimmten Viewsets ist es möglich, gezielt die Eigenschaften des Grafik–Subsystems in Bezug auf eine bestimmte Anwendung zu testen.

2.5 Das IO-Subsystem

2.5.1 ide-smart, smartsuite

Mit diesen beiden Programmen kann der Zustand von Festplatten überwacht werden. Dafür ist es nötig, dass die Festplatte und das BIOS des Rechners SMART (Self–Monitoring, Analysis and Reporting Technology) unterstützen.

2.5.1.1 Funktionen, Besonderheiten

Beide Programme unterstützen IDE–Festplatten, aber nur **smartsuite** kann auch SCSI–Festplatten untersuchen. Die Unterstützung von SCSI–Festplatten in **smartsuite** ist erst in der letzten Final–Version 2.1 des Programms implementiert. Es lässt sich damit von SCSI–Festplatten, welche dies unterstützen, die Temperatur auslesen. Jedoch scheint **smartsuite** nicht die einzelnen SMART–Werte der SCSI–Festplatten prüfen zu können. **smartsuite** gibt nur
=> `S.M.A.R.T. Sense: Okay!`
und nicht wie bei IDE–Festplatten
=> `Check S.M.A.R.T. Passed`
aus. In der Dokumentation ist nichts zu den unterschiedlichen Ausgaben des Programms vermerkt. Aus dem Quelltext des Programms ist ersichtlich, dass die einzelnen SMART–Werte der IDE–Festplatten untersucht werden. Bei SCSI–Festplatten wird jedoch nur ein Datum gelesen.

Das Programm **ide-smart** liest alle SMART–Werte aus und gibt eine Wertung ab, ob der Wert das vorgegebenen Limit erreicht hat oder nicht. Weiterhin ist zu jedem Wert vermerkt, ob es sich bei dem dazugehörigen Limit um einen Wert aus der Gruppe PreFailure oder Advisory (siehe Kapitel 1.3.5) handelt. **smartsuite** bietet dies nicht. Dafür zeigt es zu jedem Wert eine englische Kurzbeschreibung an, welches Detail der Festplatte der Wert beschreibt.

2.5.1.2 Test der Hardware

Von IDE–Festplatten, welche den ATA-3 Standard [Site-14] oder höher unterstützen, und von SCSI–Festplatten des SCSI-3 Standard [Site-15] ist das Auslesen der SMART–Informationen möglich. Alle untersuchten IDE–Festplatten unterstützten SMART. Sowohl

[34] Treiber der Grafikkarten, wobei N für den Treiber von Nvidia Version 1.0-31.23 steht und M für Mesa OpenGL 1.2 Version 3.4.2

[35] ohne Clipplane/mit Clipplane

mit **smartsuite** als auch mit **ide-smart** war es möglich, die SMART–Werte bei diesen Festplatten zu ermitteln. Es handelte sich dabei um Festplatten der Firma Western Digital ab 20 GB. Allerdings stand keine Festplatte mit entsprechenden Defekten zur Verfügung, um auch den Fehlerfall zu testen.

2.5.1.3 Begründung der Aufnahme in die Qualip-Suite

Zur Anwendung kommt das Programm **smartsuite**,da es SCSI-Festplatten unterstützt. Die Funktionalität in Bezug auf SCSI–Festplatten ist bei diesem Programm zwar wesentlich geringer als bei IDE–Festplatten, aber dies kann sich mit einer neueren Version des Programms ändern.

2.5.2 bonnie++

Mit dem Programm **bonnie++** kann der Datendurchsatz von Festplatten und Dateisystemen gemessen werden. Es wird zu jedem Test der erreichte Datendurchsatz und die Belastung der CPU in Prozent ausgegeben.

2.5.2.1 Tests des Programms bonnie++

Um den Datendurchsatz der Festplatten[36] zu testen wird folgendes ausgeführt:

- sequentielles Schreiben:

  zeichenweises Schreiben der Datei: Eine Datei wird mit der C-Standardfunktion *putc()* geschrieben. Die Schleife, welche dabei ausgeführt wird, ist klein genug, um im Instruction-Cache der CPU gehalten werden zu können. Die CPU wird durch das Ausführen des Codes der *putc()*–Funktion und durch Speicherzuweisung an die Datei belastet.

  blockweises Schreiben: Die Datei wird mit der C-Funktion *write()* geschrieben. Nur die Speicherzuweisung an die Datei belastet die CPU[37].

  Lesen und Zurückschreiben: Die Datei wird in Blöcken durch die C-Funktion *read()* eingelesen. Die Daten werden anschließend abgeändert und mit der C-Funktion *write()* in die Datei geschrieben. Hierbei werden die enthaltenen Daten überschrieben. Vor dem Schreiben der Daten ist ein zurücksetzen des Dateizeigers nötig, wofür die C-Funktion *lseek()* genutzt wird. Die CPU wird nicht mehr durch Speicherzuweisungen belastet.

- sequentielles Lesen:

  zeichenweises Lesen: Eine Datei wird mit der C-Standardfunktion *getc()* gelesen. Die dabei ausgeführte Schleife ist klein genug, um im Instruction-Cache der CPU gehalten werden zu können.

[36] Das Dateisystem wird dabei ebenfalls mit getestet, da in ein Verzeichnis einer gemounteten Partition geschrieben bzw. daraus gelesen wird. Es wird nicht auf ein Block–Gerät direkt zugegriffen.

[37] vorausgesetzt der Festplattencontroller arbeitet im DMA–Modus, sonst muss die CPU den Datentransport vom Hauptspeicher auf die Festplatte vollziehen

blockweises Lesen: Die Datei wird mit der C-Funktion *read()* gelesen.

- wahlfreier Zugriff:

  Es werden 3 Prozeduren parallel ausgeführt. Jede dieser Prozeduren setzt mit Hilfe der C-Funktion *lseek()* den Dateizeiger auf eine neue Position. Die Position innerhalb der Datei wird dabei durch die C-Funktion *drand48()* per Zufall bestimmt. Nach dem Setzten des Dateizeigers wird ein Block durch die C-Funktion *read()* gelesen. Jeder zehnte Block wird verändert und unter Verwendung der C-Funktion *write()* wieder in die Datei geschrieben. Durch die Parallelisierung kann eine höhere Auslastung erzielt werden. Es wartet immer eine Anforderung zur Positionierung des Dateizeigers auf deren Ausführung. Ziel dieses Tests ist es die Wirkung des Caches zu reduzieren, um die Zugriffszeit der Festplatte zu messen.

Falls die Größe der Testdateien mit über einem GByte festgelegt ist, so werden mehrere Dateien angelegt.

Für die Messung des Datendurchsatzes des Dateisystems wird das Anlegen, das Lesen und das Löschen von Dateien getestet. Diese drei Prozeduren werden einmal in einem sequentiellen und einmal in einem wahlfreien Test durchgeführt. Es werden Dateien angelegt, deren Bezeichnung aus sieben Ziffern und einer zufälligen Folge von alphanumerischen Zeichen besteht. Dieser wahlfreie Teil des Dateinamens besteht aus einer zufälligen Anzahl zwischen null und zwölf alphanummerischer Zeichen, die ebenfalls per Zufallsfunktion gewählt wurden. Während des sequentiellen Tests bilden sich die Dateinamen aus der fortlaufenden Nummer, gefolgt von der zufälligen Folge von alphanumerischen Zeichen. Beim wahlfreien Test beginnen die Dateinamen mit dieser zufälligen Folge.

Die Dateien werden erstellt und anschließend wird mittels der C-Funktion *stat()* auf diese zugegriffen. Die Reihenfolge dieser Zugriffe wird durch die C-Funktion *readdir()* bestimmt. Die *readdir()*–Funktion listet die Dateien im Verzeichnis auf (in der Reihenfolge, wie sie gespeichert wurden). Dies ist im Fall des sequentiellen Tests annähernd die Reihenfolge der Erzeugung. Nach dem Zugriff auf diese Dateien werden sie in ebenfalls dieser Reihenfolge gelöscht.

Falls beim Aufruf des Programms **bonnie++** per Option die maximale Größe der Dateien mit mehr als 0 Byte festgelegt wurde, werden alle Dateien mit Daten gefüllt. Der Umfang dieser Daten wird durch den Zufallsgenerator festgelegt und liegt zwischen 0 Byte und der angegebenen Grenze. Wird dann mittels der C-Funktion **stat()** auf diese Dateien zugegriffen, so wird deren Inhalt eingelesen.

2.5.2.2 Kommandozeilen-Optionen und Ausgabe

Dem Programm **bonnie++** muss, falls es durch den Benutzer *root* gestartet wird, per Option <-u> der Login–Name des Nutzers übergeben werden, unter dem das Programm ausgeführt werden soll. Das Programm ermittelt beim Start automatisch die Größe des im Rechner verbauten Hauptspeichers. Dies kann unterbunden werden durch den Parameter <-r> mit anschließender Angabe der Größe des Hauptspeichers. Wenn nicht anderweitig durch Parameter vorgegeben, nutzt **bonnie++** als Testgröße das Doppelte der Größe des Hauptspeichers. Mit dem Parameter <-s> kann die Größe der Testdatei für den Test des Datendurchsatzes der Festplatte festgelegt werden. Die Anzahl und Größe der Dateien

beim Dateisystemtest wird über den Parameter <-n> bestimmt. Durch Übergeben einer null an die Parameter <-s> bzw. <-n> werden die Tests des Datendurchsatzes der Festplatte bzw. des Dateisystems übergangen. Der Umfang der Tests des Datendurchsatzes der Festplatte lässt sich weiterhin durch den Parameter -f einschränken, indem die zeichenweisen Lese- und Schreibtests nicht durchgeführt werden. **bonnie++** zeigt auf der Kommandozeile den aktuell ausgeführten Test an. Die Ausgaben beinhalten den Namen des Rechners, die für die Tests genutzte Anzahl und Größen der Dateien, die ermittelten Werte und die bei den Tests erzeugte CPU-Last. In der letzten Zeile der Ausgabe sind alle Angaben nochmals durch Kommata getrennt aufgelistet. Dies dient der einfacheren Auswertung der Daten. Wird der Parameter <-q> übergeben, so werden bis auf diese letzte Zeile sämtliche Ausgaben an den Fehler–Ausgabestrom *"STDERR"* gesendet.

2.5.2.3 Testergebnisse

Die erzielten Ergebnisse wurden unter Nutzung des Dateisystems *XFS* und aktivierten 32 Bit- und DMA–Zugriff ermittelt. Der Testrechner war mit folgenden Systemkomponenten ausgestattet.

CPU AMD Athlon XP 2000+, 1667 MHz

Mainboard ASUS A7V266-E

Hauptspeicher 768 MByte DDR-RAM, PC2100

Festplatte Western Digital Caviar WD600BB-22CAA0

Ein Standardlauf ohne Angabe von weiteren Parametern ergab folgendes Ergebnis[38]:

```
------Sequential Output------ --Sequential Input- --Random-
-Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
11651  99 60200  23 20572   9 11112  95 48151  13 190.6   0
------Sequential Create------ --------Random Create--------
-Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
/sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
2742  25 +++++ +++  2914  19  3016  26 +++++ +++   727   6
```

Bei dem Standardlauf wurden einige Dateisystem–Tests in unter 500 ms abgeschlossen. Die Ergebnisse dieser Tests werden nicht dargestellt (statt den Ergebnissen werden mehrere "+" ausgegeben), da die Rundungsfehler mit abnehmender Testdauer immer größer werden. Deshalb wurde, abweichend zu dem Standardlauf des Programms, durch die Kommandozeilen–Optionen <-s 3g -n 70:35k> ein weiterer Versuch gestartet. Die Messung des Datendurchsatzes der Festplatte erfolgte demnach bei einer Größe der Testdateien von drei GByte. Die Leistungsfähigkeit des Dateisystems wurde mit 70*1024 Dateien bei einer Maximalgröße von 35 KByte getestet.

[38] die Ausgaben von **bonnie++** sind gekürzt

```
------Sequential Output------ --Sequential Input- --Random-
-Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
11639  98 50368  20 21350   9 11311  97 48124  13 120.0   0
------Sequential Create------ --------Random Create--------
-Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
/sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
1231  22 98736 100  1292  12  1272  23 80579 100   185   3
```

Die Gesamtgröße der Testdateien wuchs von 1.5 GByte auf 3 GByte. Somit ist der Bereich, auf den der *lseek()*–Befehl verweisen kann, ebenfalls auf das doppelte vergrößert worden. Der wahlfreie Zugriff auf Dateien wurde entsprechend verlangsamt. Das Erstellen und Löschen der Dateien wurde ebenfalls verlangsamt, da die Dateien im zweiten Test mit Daten bis zu 35 KByte gefüllt wurden. Das blockweise Schreiben wurde verlangsamt. Das Sinken des Datendurchsatzes lässt sich durch den verringerten Einfluss des Caches erklären. Der Cache wuchs bei den Tests bis auf 679 MByte[39] an. Wenn somit 1.5 GByte geschrieben werden sollen, können davon 679 MByte temporär zwischengespeichert werden. Dieses Verhältnis von zu schreibenden Daten und Cache verschiebt sich bei dem Test mit drei GByte Testdaten.

Wurde die Größe der Testdatei auf 768 MByte herabgesetzt[40], ergaben sich folgende Werte.

```
------Sequential Output------ --Sequential Input- --Random-
-Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
          95759  32 18893   7           62824  16 650.9   1
```

Die Ergebnisse für das blockweise Lesen oder Schreiben wurden erhöht, da Linux Teile der Daten im Cache halten konnte.

Weiterhin wurde ein Testlauf durchgeführt, in dem die Auswirkungen von Einstellungen verdeutlicht werden sollen, die die zur Verfügung stehende Hardware nicht zur Leistungssteigerung nutzt. Die Änderung des Zugriffs auf die Festplatte erfolgte mit dem Programm **hdparm**, welches mit den Parametern <-c0d0> aufgerufen wurde. Somit wurde der DMA-Modus deaktiviert und der Zugriff auf die Festplatte von 32 auf 16 Bit herabgesetzt. Anschließend wurde **bonnie++** ohne weitere Parameter gestartet.

```
------Sequential Output------ --Sequential Input- --Random-
-Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
3218  95  5786  55  1605  39  2522  98  3189  47 130.2  18
------Sequential Create------ --------Random Create--------
-Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
 /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
  417  95 +++++ +++   403  92   415  94 +++++ +++   255  58
```

[39] gemessen mit dem Programm **free**

[40] Dies ist nur möglich, wenn **bonnie++** mit der Option <-r> eine Größe des Hauptspeichers von kleiner oder gleich 384 MByte vorgegeben wird. Dadurch wird die automatische Erkennung der Größe des Hauptspeichers deaktiviert.

Der deutlichste Unterschied in den ermittelten Datendurchsätzen zwischen den Ergebnissen bei Standardkonfiguration und der hier verwendeten ist bei den blockorientierten Schreib- und Lese–Tests zu erkennen. Hier liegt die erzielte Datentransferrate unter 10 % der bei Standardkonfiguration erzielten Werte, wobei die CPU–Auslastung bei diesem Test mehr als das doppelte beträgt.

2.5.2.4 Begründung der Aufnahme in die Qualip-Suite

Das Programm **bonnie++** liefert reproduzierbare Ergebnisse. Es wurden Standardfunktionen verwendet, die auch auf einer anderen Hardwarebasis optimiert sind. Die auszuführenden Tests lassen sich per Parameter einschränken. Die Ausgabe des Programms ist durchdacht und eignet sich für die Auswertung durch ein übergeordnetes Programm.

2.5.3 iozone

iozone ist ein Dateisystem- und Festplattenbenchmark. Die Tests und Ausgaben des Programms **iozone** lassen sich durch 56 Parameter festlegen. Das Programm ist für einen Test von Rechnern entworfen worden, über deren Architektur (u.a. die CPU) Kenntnisse vorhanden sind und deren Leistung verglichen werden sollen. Dazu können außer der Größe der zu schreibenden oder lesenden Datei, auch die Größe der Rekords eingestellt werden, welche pro Zugriff gelesen bzw. geschrieben werden. So ist es möglich herauszufinden, bei welcher Block–Größe Engpässe des Datendurchsatzes entstehen.

Per Parameter kann die Größe des Caches und die Cache–Line–Size der CPU übergeben werden. Diese Werte nutzt **iozone** für interne Optimierungen (beispielsweise für den Zeitpunkt des Pufferabgleichs) und für das Überschreiben des Caches[41].

Die Lese- und Schreib–Tests werden wiederholt ausgeführt, um die Auswirkungen auf den Datendurchsatz zu prüfen. Die erneut gelesenen Daten könnten sich noch im Cache befinden. Bei wiederholten Schreiben einer Datei existieren die zu schreibenden Metadaten schon. **iozone** bietet an, die Lese- und Schreibtests durch alternative C–Standardfunktionen ausführen zu lassen. So kann statt *read() fread()* und statt *write() fwrite()* verwendet werden.

Es sind Optionen geschaffen, welche Daten nach Microsoft Excel exportieren und dem Testen des Zugriffs auf ein NFS-Laufwerk dienen. Dazu wird die Zeit, welche zum Schließen des Datenstroms notwendig ist, mit in die Berechnung der Transferrate einbezogen. Bei Tests von Verbindungen zwischen Rechnern, welche NFS der Version 2 und 3 nutzten, konnte jedoch kein Einfluss dieser Option auf das Ergebnis festgestellt werden.

Das Programm **iozone** wird nicht in die Qualip–Suite einfließen. Die Größe des CPU-Caches müsste ermittelt werden und **iozone** übergeben werden, da sonst nicht sichergestellt werden kann, wie sich die Einstellung einer falschen Cache–Größe auf die Ergebnisse des Tests auswirkt. Die zusätzlichen Funktionalitäten, welche **iozone** im Vergleich zu **bonnie++** bietet, finden in der Qualip–Suite keine Anwendung. Es ist nicht Ziel, die Effektivität des Caches zu bestimmen, sondern den Datendurchsatz von und zum Laufwerk zu ermitteln. Für diesen Zweck bietet **bonnie++** alle benötigten Funktionalitäten.

[41] Das Überschreiben des Caches findet nur Anwendung, wenn der Parameter <-p> übergeben wurde.

2.6 Die Netzwerkverbindung

2.6.1 netperf

2.6.1.1 Überblick

Bei **netperf** handelt es sich um einen Benchmark, der die Netzwerk–Verbindung zwischen zwei Rechnern testet. Es existiert eine Server- (**"netserver"**) und Client–Anwendung (**"netperf"**) mit. **netserver** wird entweder durch den *inetd*–Server[42] oder als Dämon[43] gestartet. Beim Start des **netperf**–Clients wird eine Verbindung zum angegebenen Server oder zu *localhost* aufgebaut. Diese Verbindung ist, unabhängig von den gewählten Tests, eine TCP–Verbindung, welche BSD–Sockets nutzt. Sie dient der Kontrolle des Tests. Es werden Informationen über den auszuführenden Test sowie Ergebnisse ausgetauscht. Diese Verbindung bleibt während der Zeit des Tests bestehen, wobei in diesem Zeitraum keine Daten über diese Verbindung übertragen werden sollten. Jedoch kann der Test durch die Kontrollverbindung auf Grund von Standardeinstellungen des Rechnersystems wie beispielsweise SO_KEEPALIVE[44] gestört werden. Falls eine TCP–Verbindung getestet werden soll, muss eine weitere Verbindung aufgebaut werden. **netperf** kann die CPU–Belastung des Client- und Server–Rechners während eines Tests messen, wobei dabei der Prozentsatz der von **netperf** verursachten CPU–Auslastung angezeigt wird. Diese Informationen werden dem "*/proc*"–Dateisystem entnommen.

2.6.1.2 Messung des Datendurchsatzes

Wenn kein anderer Test explizit per Parameter gewählt wird, misst **netperf**, wie schnell ein Datenstrom über eine TCP–Verbindung (unicast) übertragen werden kann. Die Testdauer beträgt zehn Sekunden. Alle verbindungsspezifischen Einstellungen wie beispielsweise die Puffergrößen werden bei der Standardeinstellung des Systems belassen. Der Test kann durch folgende Optionen beeinflusst werden:

- Es kann die Testdauer durch Angabe von Zeit oder zu übertragender Datenmenge festgelegt werden.
- Die Menge an Daten, welche pro Sendebefehl übertragen werden, sowie die Sende- und Empfangspuffer, sind für Client und Server wählbar.
- Es kann die Option TCP_nodelay aktiviert werden, wodurch die Wartezeiten (auf eine größere Menge zu versendender Daten) beim Versenden von TCP-Paketen abgeschaltet wird.

Dieser Test ist unter Nutzung des UDP–Protokolls durchführbar. Der dafür nötige Parameter lautet <-t UDP_STREAM>. Jedoch darf die Größe der pro Sende–Befehl zu übertragenden Daten nicht größer sein als der kleinste der beiden genutzten Puffer-Speicher (Empfangs- und Sendepuffer). Da für übertragene Pakete keine Bestätigungen versandt werden, gibt **netperf** in den Ergebnissen die gesendeten und die empfangenen Bytes aus.

[42] Server zur Verwaltung von Netzwerkdiensten [Site-11]

[43] Server–Prozess ohne angekoppeltes Terminal

[44] sendet in bestimmten Zeitabständen Daten, um eine temporär nicht genutzte Verbindung aufrecht zu erhalten

2.6.1.3 Messung der Verarbeitungsgeschwindigkeit von Anfragen

Es wird gemessen, wieviele Anfragen pro Sekunde vom Server verarbeitet und beantwortet werden können. Um diesen Test durchzuführen muss **netperf** mit dem Parameter <-t TCP_RR> aufgerufen werden. Die Anfrage- und Antwort–Pakete enthaltenen 1 Byte Daten. Der äquivalente Test unter Nutzung des UDP–Protokolls war nicht lauffähig. Er lieferte, auch nach erneuter Übersetzung von **netperf**, immer eine Anzahl der verarbeiteten Anfragen von null zurück.

2.6.1.4 Testergebnisse

Es wurde jeder Test generell mit der Option <-l 30 -c -C> gestartet, wodurch eine Testdauer von 30 Sekunden festgelegt und die Belastung der CPU von Client und Server ausgegeben wurde. Das Serverprogramm **netserver** wurde als Dämon gestartet. Die Konfiguration der für die Tests verwendeten Rechner kann der Tabelle 2.8 entnommen werden.

	Server	Client	1 GBit Client und Server
CPU	AMD K6-2+ 550MHz	AMD Athlon 750 MHz	AMD Duron 900
Hauptspeicher	512 MByte	640 MByte	256 MByte
Netzwerkkarte	3Com 3c905B-TX[45]	3Com 3c905C-TX[45]	Syskonnect SK-9843 SX[45]

Tabelle 2.8: Komponenten der Testrechner zum Test von **netperf**

Test	Daten-durchsatz	CPU-Last Server [%]	CPU-Last Client [%]
100 MBit/s vollduplex TCP [MBit/s]	93.77	29.74	15.15
100 MBit/s vollduplex UDP, 65000 Byte pro Sende-Befehl [MBit/s]	96.2	10.88	8.74
100 MBit/s vollduplex Verarbeitungsgeschwindigkeit von TCP Anfragen pro Sekunde	8659.10	36.90	32.53
100 MBit/s halbduplex TCP [MBit/s]	20.01	8.78	4.15
10 MBit/s halbduplex TCP [MBit/s]	7.09	2.36	1.47
1 GBit/s vollduplex TCP [MBit/s]	242.78	60.43	67.66

Tabelle 2.9: Testergebnisse des Programms **netperf** auf Basis unterschiedlicher Übertragungsmodi im Ethernet

Die Netzwerkkarten der Rechner waren per Switch (Extreme Networks Summit 48) durch eine Verbindung mit 100 MBit/s untereinander verbunden. Durch Einstellungen am Switch wurde diese Verbindung für den entsprechenden Test auf halbduplex beschränkt.

[45] Als Treiber für die Netzwerkkarte kamen die Kernel–Module *3c59x.o* bzw. *sk98lin.o*, welche im Kernel 2.4.19 enthalten sind, zum Einsatz.

Die Verbindung mit 10 MBit/s wurde durch Zwischenschalten eines 10 MBit Multi-Port-Repeaters erreicht, welcher das Übertragen der Daten nur halbduplex erlaubt. Bei dem Test der Verbindung mit 1 GBit/s befanden sich die Netzwerkkarten in mit 33 MHz angesteuerten 32 Bit PCI–Steckplätzen[46]. Zwischen den beiden GBit–Ethernet–Karten befand sich ein Switch Black Diamond der Firma Extreme Networks.

Die durch **netperf** ermittelten Ergebnisse sind in Tabelle 2.9 einzusehen. UDP ist verbindungslos und bietet weniger Kontrolle als TCP. Somit ist der Datendurchsatz via UDP höher als der mittels vergleichbarer Konfiguration (100 MBit/s, vollduplex) erzielte Datendurchsatz bei Nutzung des TCP–Protokolls. Die Rechner der GBit–Ethernet–Karten nutzen nicht das Potential der Karten aus, indem sie nur über 32 Bit PCI–Steckplätze verfügen. Der Datendurchsatz ist entsprechend tiefer als der theoretische Höchstwert von 1 GBit/s.

2.6.1.5 Begründung der Aufnahme in die Qualip-Suite

netperf bietet eine Testmöglichkeit der Netzwerkverbindung, welche nur die ersten vier Schichten (bis Transportschicht) des OSI–Referenzmodells nutzt. Somit ist bei Messungen des Datendurchsatzes zwischen Anwendungen, welche Daten über eine Netzwerkverbindung austauschen, eine Eingrenzung eines eventuellen Problems möglich. Der Aufbau einer Kontrollverbindung ist durchdacht. Erst dadurch wird das Testen der Übertragungsgeschwindigkeit ohne Nutzung der Anwendungsschicht möglich. Die Ergebnisse aus Tabelle 2.9 entsprechen den (auf Grund theoretischer Betrachtungen) zu erwartenden Werten. Sie sind reproduzierbar. Die Ausgabe ist durch ein übergeordnetes Programm auswertbar.

2.7 Die Softwareinstallation

Ziel ist die Überprüfung der Vollständigkeit der Installation. Da es nicht praktikabel ist, alle Dateien einzeln zu prüfen, werden für diese Aufgabe Paketmanager genutzt. Statt der Dateien wird das Vorhandensein von Paketen abgefragt.

Die beiden Arten von Linux–Programm–Paketen, die die größte Verbreitung gefunden haben, sind das von Red Hat entwickelte *rpm*- und das von Debian verwendete *deb*-Format. Für diese beiden Formate gibt es jeweils Verwaltungsprogramme, welche folgende Funktionalitäten bieten (Auszug):

- Installieren neuer Programm-Pakete
- Speichern von Informationen über installierte Pakete:
  - Name
  - Version
  - enthaltene Dateien im Paket
  - Beschreibung des Pakets

[46] Die Netzwerkkarten erlauben auch den Betrieb in 64bit/66MHz PCI–Steckplätzen, sind aber abwärtskompatibel.

- Update bereits installierter Programm-Pakete
- Zeigen des Inhalts von Paketen (Beschreibung, enthaltene Dateien)
- Löschen von Paketen
- Erstellen von Paketen

Um diese zu nutzen, existieren die Programme **rpm** und **dpkg**. Mit dem Aufruf
% rpm -qa
bzw.
% dpkg -l
lassen sich alle installierten Pakete anzeigen[47].

Allerdings variiert die Benennung der Pakete der verschiedenen Distributoren. Deshalb sollte für jede zu prüfende Distribution eine Liste von Paketen, deren Status geprüft werden soll, erstellt werden.

2.8 Suiten zum Test mehrerer Subsysteme

2.8.1 unixbench

Bei dem **unixbench**-Benchmark handelt es sich um eine Suite von mehreren Unterprogrammen, welche über ein gemeinsames Skript aufgerufen werden. Ein Ziel bei der Entwicklung dieses Benchmarks war es, die unterschiedlichen existierenden Unix-Varianten zu vergleichen. Es wird unter anderem folgendes getestet:

- der Datendurchsatz für das Kopieren von Dateien mit unterschiedlichen Puffergrößen
- die Operationen pro Sekunde bei arithmetischen Berechnung in den Bereichen Floating–Point und Integer bei variierender Größe des Datentyps
- wieviele Lese- und Schreibzugriffe auf eine Pipe (Inter–Prozess–Kommunikation) pro Sekunde möglich sind
- die Anzahl der Schleifendurchläufe für Context Switching mittels zweier Pipes (Es findet ein stetiges Wechseln zwischen zwei Pipes statt, wobei zwei Prozesse auf die Pipes abwechselnd lesend und schreibend zugreifen. Ein Prozess, welcher den Befehl erhält von einer Pipe zu lesen, blockiert, bis er erfolgreich gelesen hat.)
- wieviele Shell–Skripte pro Minute ausgeführt werden können, bei einem, acht und sechzehn Skripten, welche gleichzeitig als Hintergrundprozess gestartet werden (Die Shell–Skripte enthalten jeweils vier Befehlszeilen.)

[47] Das Programm **dpkg** zeigt jedoch die Namen der Pakete nur bis zu einer begrenzten Länge an. Um die vollen Namen zu ermitteln, müssen die installierten Pakete aus der Ausgabe von
% dpkg - -get-selections
gefiltert werden.

- Compilerdurchläufe pro Minute, wobei jeweils ein 153 Zeilen langer Quelltext zu übersetzen ist
- Schleifen pro Sekunde bei dem Dhrystone-Test (Test der Integer-Leistung der CPU, siehe [Site-12])
- CPU-Geschwindigkeit bei dem Whetstone-Benchmark in MWIPS (Millionen Whetstone Befehle pro Sekunde), wobei Floating-Point-Berechnungen durchgeführt werden (siehe auch [Site-13])

unixbench vereint somit Tests, welche viele der unterschiedlichen Komponenten eines Rechners beanspruchen.

Dass dieser Benchmark nicht in die Qualip–Suite einfließt, hat mehrere Gründe. Die arithmetischen Tests ermitteln nur jeweils eine spezielle Funktionalität der CPU. Es werden nur bestimmte arithmetische Operationen in einer Schleife ausgeführt. Der Code weist weniger Sprünge auf als beispielsweise ein Sortieralgorithmus verursachen würde. Die Wahrscheinlichkeit, ein realitätsnahes Ergebnis zu erzielen, ist geringer als zum Beispiel bei **nbench**. Ein realitätsnahes Ergebnis bedeutet in diesem Zusammenhang, dass die prozentualen Unterschiede der Leistungstests zwischen verschiedenen CPUs bei in der Praxis eingesetzten Anwendungen und entsprechenden[48] Benchmarks übereinstimmen.

Die Tests des Festplattendurchsatzes bieten einen zu geringen Funktionsumfang, da die Testgröße der zu lesenden oder schreibenden Daten nicht einstellbar ist. Dies ist jedoch entscheidend, da ein PC mit größerem Hauptspeicher mehr Daten zwischenspeichern kann, bevor diese auf die Festplatte zurückgeschrieben werden müssen.

Die Tests des C-Compilers und der Ausführungsgeschwindigkeit von Shell-Skripten schwankt je nach eingesetztem Compiler / eingesetzter Shell. Die Geschwindigkeit eines bestimmten Compilers oder einer bestimmten Shell ist für die Qualip–Suite nicht relevant.

Als weiteres Problem erweist sich die mangelnde Dokumentation. Es wird für jeden Test angegeben, wie der Dateiname des zugrunde liegenden Quelltextes heißt. Was in den Tests geschieht, ist jedoch nur den Quelltexten zu entnehmen.

unixbench eignet sich besser zum Vergleich verschiedener Unix-Betriebssysteme auf einem Rechner, als zum Vergleich verschiedener Rechner unter Linux.

2.8.2 AIM 9

AIM 9 bietet ähnlich **unixbench** Tests an, die mehrere Subsysteme des Rechners in Einzeltests prüfen. Getestet werden kann (Auszug):

CPU Operationen pro Sekunde für Addition, Multiplikation und Division von Ganz- und Gleitkommazahlen mit unterschiedlicher Anzahl von Bits pro Variable

Festplatte Datenrate für das Schreiben und Lesen von bzw. auf die Festplatte mit und ohne Synchronisation[49]

Dateisystem Operationen pro Sekunde für die Ausgabe von Verzeichnis–Inhalten sowie das Anlegen und Schließen von Dateien

[48] Art der Berechnungen (z.B. Verhältnis Floating–Point-/Integer–Berechnungen) identisch
[49] Aufruf der Funktion *sync()* nach Schreib–Befehl

Prozess Management Operationen pro Sekunde für das Ausführen der Funktionen *fork()* und *exec()*

Inter-Prozess-Kommunikation Nachrichten pro Sekunde bei Nutzung von UDP/IP, TCP/IP, shared Memory und PIPE

Algorithmen Operationen pro Sekunde bei Ausführung von Algorithmen zum Lösen linearer Gleichungssysteme, Taylor–Reihen und 3D Projektionen

Welche dieser Einzeltests ausgeführt werden sollen, wird in einer Konfigurationsdatei gewählt. Ein Script erzeugt ein Makefile, wodurch ein Aufruf des Programms **make** eine ausführbare Datei, den eigentlichen Benchmark erstellt.

Über die Konfigurationsdatei ist auch die Dateigröße für die Tests des Datendurchsatzes der Festplatten einzustellen. Somit müsste für jeden zu testenden Rechner der Wert angepasst und der Benchmark neu übersetzt werden, da die Datei die doppelte Größe des Hauptspeichers besitzen sollte, um den Cache-Effekt zu verringern. Das Programm bietet zwar an, nach den Schreibbefehlen die *sync()*–Funktion aufzurufen, doch dadurch wird eine niedrigere als die tatsächliche[50] Datenrate (durch den blockierenden Prozess) gemessen. Für die Messung des Datendurchsatzes der Festplatten ist **bonnie++** besser geeignet, da es für einen Test mit anderer Dateigröße nicht neu übersetzt werden muss und da es die CPU–Auslastung misst.

Die Inter–Prozess–Kommunikation ist von der CPU, dem Datendurchsatz des Hauptspeichers und dem verwendeten Kernel abhängig. Die CPU wird vom Programm **AIM 9** schon in anderen enthaltenen Tests untersucht. Der Datendurchsatz des Hauptspeichers wird durch **cachebench** getestet. Die Untersuchung der Geschwindigkeit des Kernels und der Inter–Prozess–Kommunikation sind nicht Teil der Qualip–Suite. Für das Prozessmanagement gilt dies ebenfalls, sie soll in der Qualip–Suite ebenfalls nicht getestet werden. Die Geschwindigkeit von Funkionen zu prüfen, welche das Prozessmanagement oder die Inter–Prozess–Kommunikation unterstützen, ist für den Vergleich mehrerer unterschiedlicher Betriebssysteme dienlich.

Die arithmetischen Tests sind nicht aussagekräftig. Die Prozessoren besitzen mehrere Pipelines zur gleichzeitigen Berechnung von Integer- und Floating–Point–Werten. Diese Möglichkeit der parallelen Abarbeitung wird durch die arithmetischen Tests nicht genutzt. Es werden ähnlich zu **unixbench** arithmetische Operationen in Schleifen ausgeführt, mit geringer Anzahl von Sprüngen. Von den fünf enthaltenen algorithmischen Tests bricht einer auf allen getesteten Rechnern mit einer Fehlermeldung ab. Die Tests der CPU sind für die Qualip–Suite schlechter geeignet als **nbench**, da dieses zehn Tests durchführt und drei Werte (Indices) zurückliefert, welche die CPU charakterisieren.

2.9 Der Stabilitätstest

2.9.1 glxgears

In dem Paket *XFree86* ist das Programm **glxgears** enthalten, was zur Demonstration der in X–Windows integrierten OpenGL–Fähigkeiten dient. Eine Beschreibung der Grafikkarten, welche OpenGL unterstützen findet sich in Kapitel 2.4.2. **glxgears** stellt drei

[50] Schreiben von Dateien, welche ein vielfaches des Hauptspeichers groß sind

rotierende Zahnräder auf dem Bildschirm dar. Es liefert dabei die Anzahl der pro Sekunde angezeigten Bilder auf dem Standardausgabestrom zurück. Das Programm startet mit einer Fenstergröße von etwa 320 x 320 Pixeln. Per Parameter können zusätzliche Informationen ausgegeben und der X-Server gewählt werden, auf dem das das Fenster erscheinen soll. Da bei diesem Test nur ein eingeschränkter Bereich der Funktionalitäten der OpenGL–Bibliothek getestet wird, eignet sich das Programm nicht zur Feststellung der Leistungsfähigkeit des Rechnersystems im Bereich OpenGL. Aber es kann in einen Stabilitätstest integriert werden. Die in Kapitel 2.9.4.2 aufgeführten Werte bestätigen dies. Bei den beiden aufgeführten Testrechnern kam ein Treiber des Herstellers zum Einsatz, wodurch die Fähigkeiten der Grafikkarten genutzt wurden. Je nach Qualität der Treiber der Grafikkarte kann die für die auf Grafikkarte und Speichersubsystem erzeugte Last somit variieren.

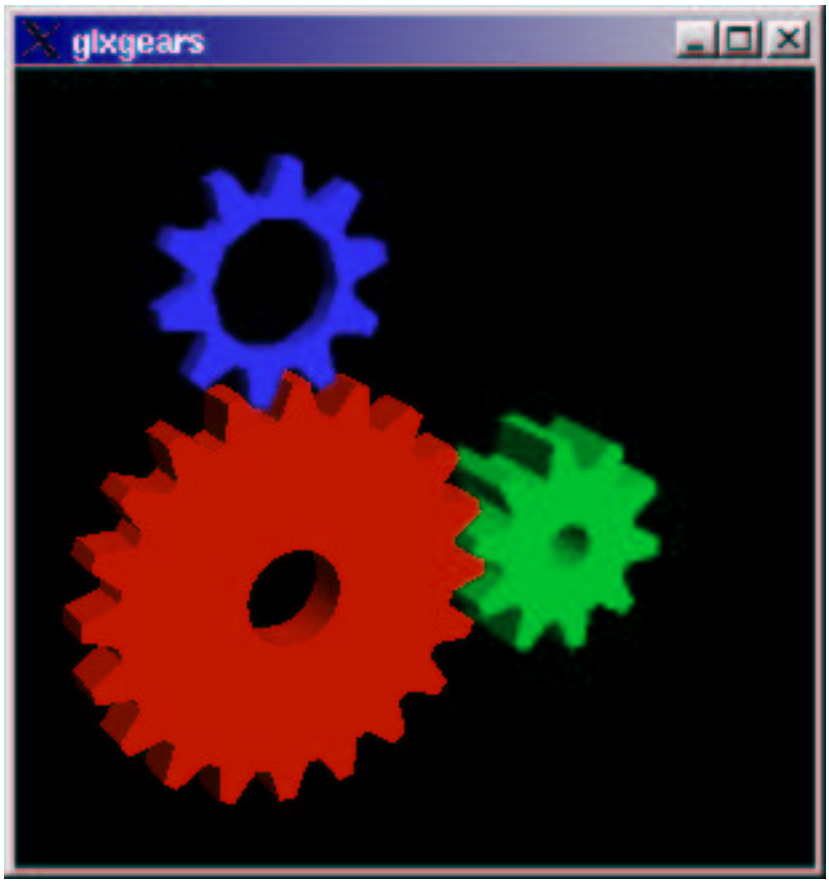


Abbildung 2.7: von **glxgears** erzeugtes Fenster, in welchem die OpenGL-Objekte dargestellt werden

2.9.2 x11perf

Dieses Programm misst die Geschwindigkeit, mit der Befehle zur zweidimensionalen Darstellung auf dem X-Server ausgeführt werden. **x11perf** testet die Geschwindigkeit für das Erstellen und Anzeigen von Fenstern, für das Anzeigen schon existierender (minimierter) Fenster und für das Neuzeichnen einer neuen Anordnung der Fenster in X-Windows.

Es werden Operationen aus dem Bereich Textverarbeitung, CAD und Bildbearbeitungsprogramme getestet. Auch Standardoperationen wie das Zeichnen von Punkten, Linien, Kreisen usw. werden getestet. Die Tests werden für eine einstellbare Dauer ausgeführt (Standard: fünf Sekunden).

x11perf wurde zur Analyse der Qualität von Grafikkarten–Treibern für X-Windows geschrieben, um die Entwickler dieser Treiber zu unterstützen. Es wird keine Funktion angeboten, durch die nur Tests ausgeführt werden, welche in der Praxis verhältnismäßig die größte Anwendung finden. Insgesamt bietet **x11perf** 285 Parameter. Unter diesen ist auch eine Option, durch welche **x11perf** alle Tests nacheinander ausführt. Jeder Test wird dabei fünf mal durchlaufen, jeweils für fünf Sekunden.

Das Programm **x11perf** ist für den Stabilitätstest geeignet, da es viele (etwa 280) unterschiedliche Tests und damit Belastungen vollführt. Falls im Stabilitätstest auch die Komponenten belastet werden sollen, welche für die zweidimensionale Darstellung genutzt werden, so bietet sich der Test an. Untersuchungen bezüglich des Stromverbrauches sind in Kapitel 2.9.4.2 verzeichnet. Allerdings ist der Stromverbrauch von nur einem Test (<-dot> - Darstellung von Punkten) gemessen worden, bedingt durch die Laufzeit aller Tests von etwa zwei Stunden.

2.9.3 lm-sensors

2.9.3.1 Überblick

lm-sensors ist ein Programmpaket zur Unterstützung von Hardware–Sensoren unter Linux. Diese Sensoren finden auf Mainboards (üblich auf Mainboards ab Baujahr 1997) und diversen Erweiterungskarten Anwendung. Die Sensor–Chips können die Spannung von einer von Chip zu Chip variierenden Anzahl von Messgebern ermitteln. Die gemessenen Spannungen müssen anschließend durch die Software umgerechnet werden. Es können die Temperaturen diverser Chips, vom Netzteil gelieferte Spannungen und Drehzahlen von Lüftern überwacht werden. Es wurde kein Programm gefunden, welches vergleichbare Funktionalitäten aufweist. Die Hersteller der Mainboards liefern zwar Software mit, welche ähnliches leistet, doch ist diese nur mit Microsoft–Betriebsystemen kompatibel. Somit wird **lm-sensors** in der Qualip–Suite Anwendung finden.

2.9.3.2 Kommunikation mit den Sensor-Chips

Die Kommunikation mit den Sensoren erfolgt entweder über den ISA-Bus oder über den SMBus. Der SMBus (“System Management Bus”) ist ein serieller Bus, welcher über zwei Kommunikationskanäle verfügt. Er dient zur Überwachung und Steuerung der Hardware. Der SMBus basiert auf dem I2C-Bus. Somit können SMBus-Chips und I2C-Chips an den selben (I2C-)Bus gekoppelt werden. Für die Verwaltung dieses Busses ist ein Steuer–Chip verantwortlich. Jeder Sensor–Chip, welcher an diesem Bus angeschlossen ist, wird über eine sieben Bit Adresse vom Steuer–Chip angesprochen. Zur Ansteuerung der Steuer- und Sensor–Chips wird jeweils ein Kernel–Modul benötigt[51]. Ist der Sensor–Chip an den ISA-Bus gekoppelt, so ist zwar kein Kernel–Modul für den Steuer–Chip zu laden, jedoch

[51] Kernel–Module sind dynamisch einbindbare Codesegmente des Systemkerns, welche bestimmte Funktionalitäten zur Verfügung stellen.

ist außer dem Kernel–Module für den Sensor–Chip selber noch eines für die Umsetzung von ISA- zum I2C-Bus notwendig. Die Sensor-Chips lesen die Daten alle 1.5 Sekunden[52] neu ein.

2.9.3.3 Konfiguration des Programms

Das Programm **lm-sensors** benötigt Kernel–Module[53], um Abfragen der Sensoren zu ermöglichen. Diese Module werden im Quellcode–Paket von **lm-sensors** mit geliefert und müssen auf dem Zielrechner mit einem C–Compiler übersetzt werden. Dazu müssen auch die Kernel–Quellen des aktuell laufenden Kernels auf dem Rechnersystem vorhanden sein. Sind die Kernel–Module erstellt, so kann durch Starten des Programms **sensors-detect** ermittelt werden, welche Sensor-Chips auf dem System vorhanden sind. Die Ausgabe von **sensors-detect** enthält eine kurze Anleitung, wie die Konfigurationsdateien zu modifizieren sind, bzw. welche Module geladen werden müssen.

Wenn **sensors-detect** unterstützte Sensor–Chips im Rechnersystem erkannt hat, können anschließend mit dem **sensors**, welches ebenfalls im Programmpaket enthalten ist, die Sensor–Chips abgefragt werden. Die ermittelten Werte schwanken je nach Hersteller und Typ des Motherboards bzw. der Erweiterungskarten. So können die ermittelten Werte trotz gleicher Sensor–Chips variieren. Beispielsweise ist die Anzahl der Kontrollzeichen[54] der Lüfter im Rechnersystem, welche pro Umdrehung durch den Lüfter erzeugt wird, je nach Lüfter eins oder zwei.

Durch diese Besonderheiten jedes Rechnersystems muss die Datei "*/etc/sensors.conf*" angepasst werden, um die genauen Werte der Hardware–Sensoren zu ermitteln. In dieser Datei sind für jeden Chip Angaben enthalten, wie die gemessenen Daten umzurechnen sind. Dabei folgen die Berechnungen der Temperaturen dem Schema $y = mx+n$. Hierbei ist *x* der vom Sensor–Chip gelieferte Wert und *y* der Wert, welcher ausgegeben werden soll. Um somit *m* und *n* zu ermitteln, sind zwei unterschiedliche Temperaturen zu erzeugen und mit einem mit Mitteln des Herstellers des Mainboards abgelesenen Wert (aus dem BIOS abgelesener Wert) abzugleichen.

2.9.4 cpuburn

2.9.4.1 Überblick

Dieses Programm wurde von Robert Redelmeier zu dem Zweck verfasst, seine CPU nach vorherigem Übertakten auf Stabilität zu überprüfen. Dazu sollte die CPU so gut wie möglich ausgelastet werden. Als Maß für die Last, unter welcher eine CPU betrieben wird, wählte er die Stromstärke, welche durch die CPU fließt und die Temperatur der CPU. Es werden nicht alle Teilbereiche der CPU ausgelastet, dazu sind heutige CPUs zu komplex. Es wurde gezielt nach Befehlen gesucht, welche die höchste Stromstärke erzeugen und entsprechend Hitze produzieren. **cpuburn** wurde komplett in Assemblercode verfasst. Somit ist es möglich, gezielter Anweisungen auszuführen und Register zu nutzen, ohne dass der Compiler dies ändert. In der aktuellen Version 1.4 umfasst **cpuburn** sechs Programme.

[52] Dies ist ein üblicher Wert, welcher je nach verwendeter Hardware variieren kann.

[53] Dies sind die i2c- und **lm-sensors**–Kernel–Module.

[54] ticks, Spannungs-Hoch bzw. -Tief, welches durch Eintreten einer bestimmten Situation ausgelöst wurde

1. **burnP5**: Test von Intel Pentium-Prozessoren und kompatiblen
2. **burnP6**: Test von Intel Pentium Pro-Prozessoren und kompatiblen
3. **burnK6**: Test von AMD K6-Prozessoren
4. **burnK7**: Test von AMD Athlon-Prozessoren
5. **burnBX**: Test von CPU, Northbridge und Hauptspeicher bei Mainboards mit Intel BX-Chipsatz
6. **burnMMX**: Test von CPU, Northbridge und Hauptspeicher bei Prozessoren, welche die MMX-Erweiterungen beinhalten

2.9.4.2 Test auf zwei Hardware-Plattformen

Der für die Messungen genutzte Rechner 1 hatte folgende Spezifikationen:

CPU: AMD Athlon XP2100+, 1733MHz

Mainboard: Asus A7V333

Hauptspeicher: 768 MB DDR-RAM mit 166 MHz Taktfrequenz

Grafikkarte: Elsa Gladiac 511 MX mit AGP-Interface

Laufwerke: Western Digital WDC WD800JB-00CRA1, Toshiba DVD-ROM SD-M1612, Diskettenlaufwerk der Firma Alps

Netzteil: Enermax EG365AX-VE(G)

Rechner 2 unterscheidet sich nur in drei Komponenten von dem, mit welchem die ersten Messungen durchgeführt wurden:

CPU: Intel Pentium 4, 2400 MHz, 133 MHz Frontside-Bus

Mainboard: Asus P4T533-C

Hauptspeicher: 1024 MB RIMM PC 800

Es sollte die Wirksamkeit der **cpuburn**–Programme mit anderen getesteten Programmen verglichen werden. Hierzu wurde versucht, dies über eine Messung der Temperatur der Chips zu realisieren. Jedoch war es bei dem Testrechner 2 mit dem Mainboard Asus P4T533-C nicht möglich, die Temperatur der CPU unter Linux auszulesen[55]. Um die Temperatur von Grafikchips zu ermitteln wurden Temperatursensoren mit Wärmeleitpaste in die Kühlkörper der Grafikchips eingebracht. Doch wurde hierbei neben der Temperatur des Grafikchips auch die Temperatur der Luft im Computergehäuse gemessen. Ein Messgerät, welches via Infrarot die Temperatur misst, stand nicht zur Verfügung.

Nachdem Versuche, die Temperatur der CPU, Northbridge- und Grafikchips zu ermitteln fehlgeschlagen sind, wurde versucht, die erzeugte Last der Testprogramme anhand

[55]Die zum Testzeitpunkt aktuellste Version 2.6.5 des Programms **lm-sensors** war dazu nicht in der Lage.

deren Strom–Verbrauchswerten zu ermitteln. Dazu wurden die Primär- und die Sekundär–Spannung der Netzteile gemessen.

Die Leistung am Eingang des Netzteils (Primär–Leitung, 230 V Wechselspannung) wurde mit dem Messgerät EKM 265 aufgezeichnet. Die Toleranz des Gerätes bei Messungen beträgt +/- (1 % + 0.3 W). Das Gerät ermittelt die Wirkleistung des Stromverbrauchers. Zur Berechnung der Stromstärke müsste die Phasenverschiebung[56] bekannt sein. Die gemessene Wirkleistung ist auf Grund der Messmethode nicht exakt. Das Gerät müsste die Leistung mit hoher Frequenz (wie z.B. 100 kHz) abtasten. Bei dem Gerät wird dies allerdings nicht der Fall sein (die Bedienungsanleitung gibt keinen weiteren Aufschluss diesbezüglich). Es ist zu erwarten (da in der Beschreibung des Gerätes diese Eigenschaft nicht angeführt wurde und auf Grund der Preisklasse des Gerätes), dass das Gerät die Mittelwerte der Spannung und Stromstärke multipliziert [c't_21/1999].

Zur Messung der Stromstärke der Sekundär–Leitungen (3.3 V, 5 V und 12 V) wurde ein Zangenamperemeter Voltcraft 135 genutzt. Es wurden nur diejenigen Sekundärleitungen in die Messung einbezogen, welche das Mainboard mit Strom versorgen. Der Strom, welcher durch die in Rechner 1 und 2 integrierten Laufwerke floss, wurde bei der Messung an der primären 230 V–Leitung des Netzteils berücksichtigt, während dieser bei den sekundären Leitungen nicht mit in das Ergebnis einfloss. Die Messtoleranzen sind für den verwendeten Messbereich mit +/-(2 % + 0.3 A) angegeben. Es wurden Probemessungen durchgeführt, bei denen zusätzlich zum Zangenamperemeter ein 8.29 Ω Widerstand, eine Conrad PS 303-D Stromversorgung und ein Voltcraft VC 608 Multimeter zum Einsatz kamen. Das Multimeter und der Widerstand wurden in Reihe geschalten. Der Innenwiderstand des Multimeters bei Strommessungen beträgt 0.15 Ω, die Messleitungen des Multimeters hatten zusätzlich einen Widerstand von 0.10 Ω. Es wurde die Stromstärke gemessen, welche bei einer an der Stromversorgung eingestellten Spannung von 5 V und 8.5 V fließt. Für jede dieser beiden Spannungen wurden fünf Messungen mit dem Zangenamperemeter durchgeführt. Vor jeder Messung wurde ein Nullabgleich des Zangeamperemeters durchgeführt. Die Stromversorgung liefert eine konstante Spannung. Sie zeigt zusätzlich die Stromstärke an, welche in der Tabelle 2.10 angegeben wurde.

Messung	Multimeter	Zangenamperemeter	Stromversorgung
bei 5 V	0.589 A	min:0.54 A; max:0.60 A ø 0.57 A	0.58 A
bei 8.5 V	1.000 A	min:0.92 A; max:1.01 A ø 0.97 A	0.98 A

Tabelle 2.10: Probemessungen mit Zangenamperemeter und Multimeter zur Prüfung der Genauigkeit

Grund für die ermittelten Schwankungen bei den Messungen mit dem Zangenamperemeter könnte, außer der Messtoleranz, der verschobene Nullpunkt vor und nach einer Messung sein. Ergebnis dieser Voruntersuchung ist, dass alle Messungen mehrfach durchgeführt und die Mittelwerte der Ergebnisse errechnet werden müssen. Der Versuch die Stromstärke mit Hilfe eines genaueren Messgeräts zu bestimmen schlug fehl. Wenn der Stromfluss der 3.3 V - Leitung über das Multimeter Voltcraft VC 608 geleitet wur-

[56] Winkel der Verschiebung der Kurve der Stromstärke zu der Kurve der Spannung

de, so lies sich der Rechner nicht mehr starten. Dies ist eventuell auf den Widerstand des Messgeräts von 0.25 Ω zurückzuführen (0.15 Ω Innenwiderstand und 0.1 Ω Widerstand der Messleitungen). Eine Umgehung dieses Problems durch Messung des Spannungsabfalls über einen Präzisionswiderstand würde ebenfalls zu Ungenauigkeiten führen. Der Präzisionswiderstand müsste hierfür einen geringeren Widerstand als das Multimeter bei Messungen der Stromstärke aufweisen. Bei diesem geringen Widerstand ist der Spannungsabfall ebenfalls nahe der Messgenauigkeit der Multimeters.

Alle sekundären Leitungen des Netzteiles, welche unterschiedliche Spannungen lieferten wurden separat gemessen. Dabei wurde Sorge getragen, dass die Bedingungen bei jeder Messung möglichst gleich waren. Jedoch können sich elektrische Eigenschaften bzw. die Stromstärke von Temperatursensoren, schneller drehenden Lüftern und den zu messenden Komponenten selber durch Aufheizung des Rechners ändern. Zu verlässlicheren Ergebnissen wäre eine Ausrüstung notwendig gewesen, welche es erlaubt, die Stromstärken aller sekundären Leitungen des Netzteils gleichzeitig zu messen. Somit wird die Toleranz der Werte auf +/- 20 % festgelegt.

Die Ergebnisse der an Rechner 1 durchgeführten Messungen können in Tabelle 2.11 eingesehen werden, die Ergebnisse des Rechners 2 in Tabelle 2.12.

	3.3V, sekundär	5V, sekundär	12V, sekundär	230V, primär
Run–Level 1	5.62 A	9.96 A	0.56 A	118.7 W
Run–Level 2 mit X-Oberfläche	5.86 A	10.14 A	0.56 A	120.2 W
burnK7	5.67 A	17.83 A	0.56 A	165.1 W
burnMMX mit 64 KByte Testgröße	5.72 A	15.98 A	0.56 A	133.9 W
burnMMX mit 64 MByte Testgröße	6.50 A	11.96 A	0.56 A	152.9 W
nbench	5.7 A	13.97 A	0.56 A	144.9 W
x11perf -dot	6.59 A	12.73 A	0.56 A	139.1 W
viewperf	6.93 A	12.34 A	0.56 A	140 W
glxgears	6.41 A	14.12 A	0.56 A	147.9 W

Tabelle 2.11: Stromstärke der auf das Mainboard führenden sekundären Leitungen des Netzteils und Leistung der primären Leitung des Netzteils der Testprogramme auf Rechner 1

Die Programme **burnMMX**, **burnK7**, **burnP6** wurden auf den Rechnern aus Run–Level 1 heraus gestartet, ohne den X–Server und Netzwerkdienste. Die Programme **glxgears**, **x11perf** und **viewperf** wurden aus Run–Level 2 heraus gestartet. Es wurden somit außer den Testprogrammen noch zusätzlich die Programme **kdm**, **kde2** und der X-Server ausgeführt.

Bei Rechner 1 stellten sich an der 12V - Leitung bei sämtlichen Testprogrammen keine messbaren Unterschiede ein. Bei diesem Rechner scheinen nur Laufwerke und Lüfter durch die 12V - Leitung versorgt zu werden. Bei Rechner 2 wurde der Prozessor, nach Maßgabe von Intel um die Stromstärke[57] zu senken, durch die 12 V - Leitung versorgt.

[57] durch die die sekundären Leitungen des Netzteils belastete werden

Testprogramm	3.3V, sekundär	5V, sekundär	12V, sekundär	230V, primär
Run–Level 1	5.15 A	1.58 A	1.21 A	78.1 W
Run–Level 2 mit X-Oberfläche	5.12 A	1.68 A	1.23 A	79.2 W
burnP6	5.02 A	1.54 A	4.80 A	129.0 W
burnMMX mit 64 KByte Testgröße	4.92 A	1.59 A	4.01 A	117.3 W
burnMMX mit 64 MByte Testgröße	5.87 A	1.85 A	3.93 A	122.9 W
nbench	5.02 A	1.71 A	3.94 A	118.8 W
x11perf -dot	5.60 A	1.68 A	4.00 A	120.5 W
viewperf	5.91 A	1.67 A	3.18 A	113.6 W
glxgears	5.46 A	1.62 A	4.50 A	126.3 W

Tabelle 2.12: Stromstärke der auf das Mainboard führenden sekundären Leitungen des Netzteils und Leistung der primären Leitung des Netzteils der Testprogramme auf Rechner 2

Das Programm **burnK7** bzw. **burnP6** bei Rechner 2 erwirkt die höchsten gemessenen Stromstärken auf der 5 V bzw. 12 V - Leitung. Die CPU des Rechners 1 wird aus der 5 V - Leitung des Netzteils versorgt.

Den Speicher und die Northbridge einer bestimmten Leitung des Netzteils zuzuordnen ist nicht möglich. Bei einer Erhöhung der Testgröße des Programms **burnMMX** von 64 KByte auf 64 MByte wird außer dem Hauptspeicher auch die Northbridge belastet, welche die Daten zwischen CPU und RAM transportiert und puffert. Bei Rechner 1 war ein Anstieg in der 3.3 V und ein Abfall der Stromstärke in der 5 V - Leitung zu verzeichnen. Bei Rechner 2 stieg die Stromstärke der 3.3 V und 5 V - Leitung. Die Stromstärke der 12 V Leitung fiel dagegen. Somit wird bei einer Testgröße von 64 MByte der Prozessor weniger belastet als bei einer Testgröße von 64 KByte. Dies kann nur dadurch erklärt werden, dass der Prozessor die Daten schneller verarbeiten könnte, als sie zugreifbar sind. Der Prozessor muss Wartezyklen ausführen.

Die Komponenten der Rechner werden mit folgenden Spannungen versorgt:

CPU: 1.75 V (Rechner 1), 1.5 V (Rechner 2)

Grafikkarten: 2.05 V

Hauptspeicher: 2.5 V (Rechner 1), 1.4 V +/- 0.4 V (Rechner 2)

North- u. Southbridge: unbekannt (Rechner 1), 1.8 V (Rechner 2)

PCI-Karten: 5 V

Laufwerke: 5 und 12 V

Außer der für die PCI-Karten und Laufwerke liegen alle hier aufgeführten Spannungen unter den drei unterschiedlichen Spannungen, welche das Netzteil liefert. Eine Zuordnung

ist somit auf diese (spekulative) Weise auch nicht möglich, da beispielsweise keine Komponente direkt mit 3.3 V betrieben wird. Einzig zuordenbar ist die Stromversorgung der CPU.

Bei Rechner 2 wird die CPU durch die 12 V - Leitung mit Strom versorgt. Für die CPU, welche in Rechner 2 eingesetzt ist, wird einer Maximallast von 57.8 W ausgewiesen. Dies entspricht einer Belastung von von 4.82 A der 12 V - Leitung des Netzteils. Der gemessene Unterschied der Stromstärke zwischen Run–Level 1 und der Ausführung von **burnP6** beträgt 3.59 A. Die CPU des Rechner 1 wird durch die 5 V - Leitung mit Strom versorgt. Die in Rechner 1 verbaute CPU hat eine Maximallast von 72 W, was einer Belastung von 14.40 A der 5 V - Leitung des Netzteils entspricht. Zwischen Run–Level 1 und der Ausführung von **burnK7** beträgt der gemessene Unterschied der Stromstärke 7,87 A. Bei beiden Rechnern kann allerdings nicht beurteilt werden, welche Leistung die CPU bei Run–Level 1 benötigt. Auch verbraucht die Umwandlung der Spannung von 5 V zu 1.75 V bzw. 12 V zu 1.5 V, also von Sekundärspannung des Netzteils zu der von der CPU benötigten Spannung, Leistung. Diese ist von dem verwendeten Mainboard abhängig. Der Stromverbrauch dieser Transformation der Spannung wird aber mit steigender Stromstärke, die durch die CPU fließt, ebenfalls steigen.

Das in den Testrechnern verbaute Netzteil Enermax EG365AX-VE(G) spezifiziert die maximalen Stromstärken an den jeweiligen Leitungen mit 32 A für die 3.3 V und 5 V - Leitung, 26 A für die 12 V - Leitung. Somit sind auf allen Leitungen des Netzteils bei den durchgeführten Tests bei Rechner 1 maximal 55.7 % und bei Rechner 2 maximal 18.5 % der Stromstärke erreicht, welche das Netzteil liefern kann.

Bei den Testprogrammen wurden die Laufwerke nicht berücksichtigt, da kein Programm auf diese zugreift. Es wurden auf Rechner 1 bei dem DVD-ROM–Laufwerk und der Festplatte die Stromstärken der diversen Betriebszustände gemessen. Die Ergebnisse dieser Untersuchung sind in Tabelle 2.13 angegeben. Da in Rechner 2 die selbe Festplatte und das selbe DVD-ROM–Laufwerk genutzt werden, sind diese Werte übertragbar.

Betriebs-zustand	Festplatte 5 V - Leitung	Festplatte 12 V - Leitung	DVD-ROM-LW[58] 5 V - Leitung	DVD-ROM-LW[58] 12 V - Leitung
kein Zugriff	0.58 A	0.25 A	0.31 A	0.1 A
Lesen	0.92 A	0.22 A	0.59 A	0.78 A
Schreiben	0.78 A	0.24 A	-	-

Tabelle 2.13: Stromstärke der sekundären Leitungen des Netzteils von DVD-ROM- und Festplatten-Laufwerken

Wenn beide Laufwerke in beiden Rechnern genutzt werden, so würde sich die maximale Auslastung einer beliebigen Leitung der Netzteile wie folgt ändern: Der Rechner 1 würde maximal 60.4 %, Rechner 2 maximal 22.4 % der Stromstärke erreichen, welche das Netzteil liefern kann. Es ist ersichtlich, das die Verteilung der Stromverbraucher auf die sekundären Leitungen des Netzteils bei Rechner 2 besser gelingt als bei Rechner 1. Durch das großzügig dimensionierte Netzteil sind bei Rechner 1 dennoch keine Stabilitätsprobleme zu erwarten. Allerdings kann die Belastung des Netzteils durch eine Kombination

[58] DVD-ROM-Laufwerk

der Belastungen und kurzzeitige Spitzenströme (Anlaufen des DVD-ROM–Laufwerks) auch höhere Werte erreichen.

2.9.4.3 Begründung der Aufnahme in die Testsuite

cpuburn ist ein geeignetes Programm, um die Anfälligkeit der Hardware gegenüber Instabilitäten zu testen. Die Strom–Messungen weisen das Programm als das dafür am besten geeignete aller untersuchten Testprogramme aus. Bei der Ausführung von **burnP6** bzw. **burnK7** wurden die höchsten Stromstärken auf der 12–Volt- bzw. 5–Volt–Leitung des Netzteils gemessen. Da das Programm laut Autor nur auf dem Prozessor Last erzeugt, ist dieses Programm am besten geeignet, um zu prüfen, ob die Kühlung des CPUs ausreichend dimensioniert ist und die Stromversorgung gewährleistet ist.

Ist die Kühlung nicht nicht hinreichend dimensioniert, so wird der Prozessor entweder gedrosselt[59] oder dieser wird über seine Spezifikation hinaus erhitzt. Dies kann zu einem Absturz[60], zu dauerhafter Beschädigung oder zu entsprechenden Warnsignalen[61] des PC's führen.

[59] entsprechendes Mainboard bzw. CPU vorausgesetzt

[60] Der CPU Pentium 3 der Firma Intel wird bei einer bestimmten durch den internen Temperatursensor ermittelten Temperatur abgeschaltet. Dies führt zu einem Stillstand des Rechnersystems.

[61] Die im BIOS eingestellten Grenzenwerte für Temperaturen und Rotationsgeschwindigkeiten, welche zu einer Warnmeldung des Rechners per PC–internen Lautsprecher führen sollten, sind unter Linux nicht wirksam. Sie müssen durch lm-sensors abgefragt werden und durch ein Programm überwacht werden.

Kapitel 3

Qualip – Testsuite

3.1 Verwendete Programmiersprache

Die Sprachen Java, C, C++, Shellscript und Perl sind zur Umsetzung der Qualip–Suite in Betracht gezogen worden. Um den Nutzern der Software eine Erweiterung des Programms einfach zu ermöglichen, bieten sich Interpretersprachen an, da der Nutzer das Programm nach einer Änderung nicht neu übersetzen muss. Interpretersprachen haben den weiteren Vorteil, dass damit programmiertes seltener abstürzt (allgemeiner Erfahrungswert, keine Pointerarithmetik, auch sind Fehler wie ein Buffer Overflow nicht möglich). Die Sprache sollte einen möglichst einfachen Umgang mit Dateien und Werkzeuge zum Durchsuchen von Zeichenketten bieten.

Perl ist eine Interpretersprache. Es ist verbreitet und somit eventuell[1] auf dem Zielrechner schon vorhanden - muss also nicht erst installiert werden. Die Entwicklungszeit ist kurz im Vergleich zu C, C++ und Java [Preche]. Perl ist geschaffen worden, “um die einfachen Aufgaben einfach zu halten, ohne die schwierigen Aufgaben unmöglich zu machen” [WaChOr]. Es eignet sich zum Ausführen anderer Programme. Durch so genannte Formate wird es erleichtert, die Ausgabe der einzelnen Testprogramme einheitlich zu gestalten.

3.2 Struktur

Es werden für alle Teil–Programme der Suite folgende Regeln eingehalten:

1. Der Dateiname der für einen Test auszuführenden Datei endet auf “.run”.

2. Die in dem Programm verwendeten Regular Expressions werden in Dateien mit der Endung “.extract” abgelegt. Dies schafft Übersichtlichkeit in der auszuführenden Datei[2], vor allem bei mehrfacher Verwendung derselben Regular Expression.

3. Die Ausgaben eines Testprogramms werden in eine Datei, deren Name auf “.out” endet, abgelegt. Dies ist bei einer eventuellen Fehlersuche nützlich.

[1] Bei der Standardinstallation aller getesteten Distributionen wird es installiert.

[2] die Regular Expressions, welche in der Qualip–Suite verwendet werden sind bis zu 95 Zeichen lang

4. In der Datei, deren Name auf ".soll" endet, werden die Vergleichswerte abgelegt. Die Struktur dieser Datei ist vom Testprogramm abhängig.

5. Die extrahierten Ergebnisse und die Wertung[3], werden in einer Datei mit der Endung ".ist" abgelegt. Der Inhalt dieser Datei ist durch Nutzung eines Formates einheitlich.

3.3 Teilprogramme der Suite

3.3.1 Module

Mehrfach genutzte Funktionen sind in Modulen (Bibliotheken) eingebracht worden. Dabei wurden die Funktionen auf zwei Module verteilt. Im Modul *qualipformat.pm* sind Funktionen enthalten, welche die Ausgabe in einem einheitlichen Format ermöglichen und den Umgang mit den speziellen[4] Dateien der Qualip erleichtern. Das Modul *qualipfunc.pm* beinhaltet Funktionen, welche Systemeigenschaften ermitteln und Systemaufrufe für den Nutzer ausführen.

3.3.2 Qualip::bonnie

In diesem Unterprogramm der Qualip–Suite wird **bonnie++** so ausgeführt, dass der Datendurchsatzes des Dateisystems und das zeichenweise Schreiben und Lesen nicht getestet wird. Wenn der Zugriff auf die Festplatte nicht unter Nutzung der leistungssteigernden Möglichkeiten wie z.B. dem DMA–Modus erfolgt, wirkt sich dies am deutlichsten auf den blockweise ausgeführten Schreib- und Lesetests aus. Dementsprechend wird **bonnie++** mit den Parametern <-f -n 0> aufgerufen.

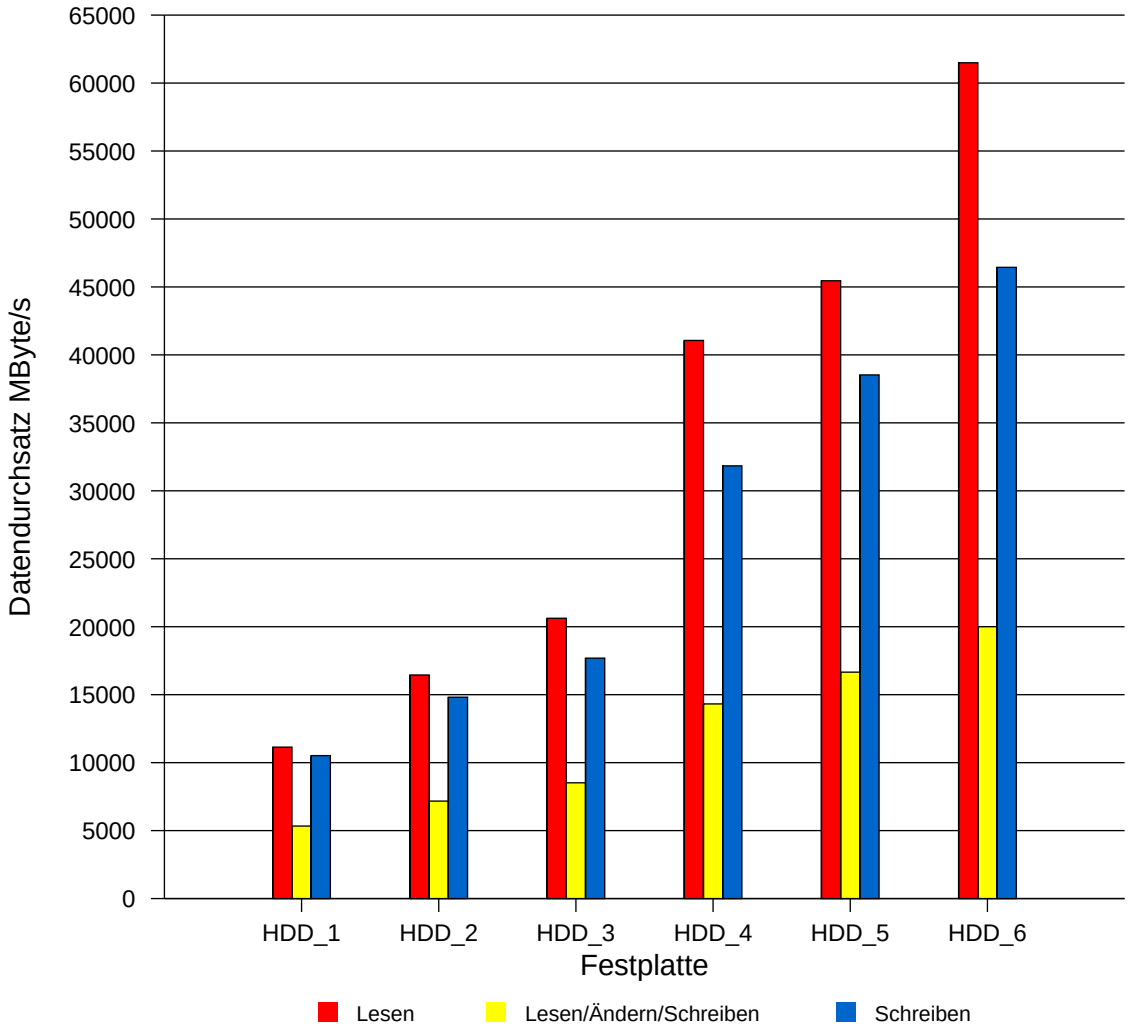
Als Maß für die Sollwerte wäre eine Datenbank mit den jeweiligen Geschwindigkeiten der Festplatten die präziseste Vergleichsmöglichkeit. Jedoch würde diese Datenbank ständig gepflegt werden müssen. Sämtliche Festplatten in möglichst unterschiedlichen Rechnern müssten vorher getestet werden, um durch Vergleichsmessungen die Sollwerte festlegen zu können. Da dies nicht praktikabel ist, wurde nach anderen Möglichkeiten gesucht.

Maßgeblich entscheidend für die Zugriffsgeschwindigkeit und den Datendurchsatz einer Festplatte sind deren Cache, die Anzahl der Umdrehungen pro Minute und die Datendichte.

Für den blockweisen Datendurchsatz ist der Cache der Festplatte in geringem Maße entscheidend. Es wurden Messungen mit zwei Festplatten der Firma Western Digital durchgeführt. Die Bezeichnungen der Festlatten sind WDC WD800BB-00CAA1 bzw WDC WD800JB-00CRA1. Die Festplatten gleichen sich in der Datendichte und in der Anzahl der Umdrehungen und unterscheiden sich in der Größe des Caches von 2 und 8 MByte. Die mittels **bonnie++** bei den Festplatten ermittelten Ergebnisse sind bei der Festplatte mit dem größeren Cache jedoch im Mittel nur um 4 % (blockweiser Schreib- und Lesetest) höher.

[3] ob der Test bestanden wurde (Vergleich mit den Sollwerten)

[4] Dateien zu den Unterprogrammen der Qualip, welche auf ".soll", ".extract", ".ist" und ".out" enden



Für die Tests verwendete Festplatten:

HDD_1: Seagate ST34520N, Ultra−Wide−SCSI−Interface, 4.5 GByte Kapazität
7200 Umdrehungen pro Minute, 0.5 MByte Cache

HDD_2: IBM DGHS09Y, Ultra−Wide−SCSI−Interface, 9 GByte Kapazität
7200 Umdrehungen pro Minute,1 MByte Cache

HDD_3: IBM DNES−318350, Ultra−2−Wide−SCSI−Interface, 18 GByte Kapazität
7200 Umdrehungen pro Minute, 2 MByte Cache

HDD_4: Western Digital WDC WD300BB−32AUA1, IDE−Interface, 30 GByte Kapazität
7200 Umdrehungen pro Minute, 2 MByte Cache

HDD_5: IBM IC35L040AVER07−0, IDE−Interface, 40 GByte Kapazität
7200 Umdrehungen pro Minute, 2 MByte Cache

HDD_6: Western Digital WDC WD800BB−00CAA1, IDE−Interface, 80 GByte Kapazität
7200 Umdrehungen pro Minute, 2MByte Cache

Abbildung 3.1: **bonnie++**-Messungen verschiedener Festplatten bei gleichen Testbedingungen

Die Anzahl der Umdrehungen pro Sekunde ist nicht dem "*/proc*"–Filesystem entnehmbar. Es müsste eine Datenbank aus Festplatten und entsprechender Anzahl an Umdrehungen pro Sekunde aufgebaut werden. Auch dies ist nicht praktikabel. Der Sollwert wird somit anhand der Größe der Festplatte bestimmt, da sich dies in gewissen Grenzen auf die Datendichte übertragen lässt. Hierzu wurden Messungen durchgeführt, deren Resultate in Abbildung 3.1 einzusehen sind. Die Ergebnisse wurden auf dem jeweils gleichen Rechnersystem mit unterschiedlichen Festplatten und dem Aufruf
% bonnie -f -n 0
erzielt.

Die Komponenten des Testrechners sind in Tabelle 3.3.2 verzeichnet Alle Festplatten

CPU	AMD Duron 750 MHz
Mainboard	ASUS A7V133
Hauptspeicher	256 MByte SD-RAM
SCSI-Controller	Adaptec 2940 UW

Tabelle 3.1: Komponenten der Rechner für den Test unterschiedlicher Festplatten mit **bonnie++**

wurden zum Test als einziges Laufwerk am jeweiligen Kanal (Kabel) angeschlossen. Das Dateisystem der Partition, in denen die Tests ausgeführt wurden, war XFS.

Das Unterprogramm *Qualip::bonnie* ermittelt pro Festplatte die Partition, in deren Grundverzeichnis geschrieben werden kann. Von diesen Partitionen wird pro Festplatte jeweils die Partition mit **bonnie++** getestet, welche den meisten freien Speicherplatz bietet. Voraussetzung ist, dass der freie Speicherplatz mindestens die doppelte Größe des Hauptspeichers aufweist. Für jede dieser Festplatten werden die Sollwerte anhand deren Größe bestimmt und die erreichten Werte entsprechend bewertet.

3.3.3 Qualip::cachebench

Es sind von der Firma AMD die CPUs K6, Athlon, Duron, Athlon XP verfügbar und von der Firma Intel der Pentium 2, 3 und 4, bzw. deren Celeron-Versionen. Alle CPUs eines Herstellers sind durch die Angaben CPU Family, Modell und Stepping genau identifiziert. Ab CPU Family sechs[5] ist generell bei allen AMD- und INTEL–Prozessoren der 2nd-Level Cache im CPU integriert. Beide Firmen bieten ab dieser CPU Familie sechs Prozessoren in unterschiedlichen Preis–Segmenten an. Die günstigeren Prozessoren waren dabei mit einem Viertel oder der Hälfte an 2nd-Level Cache ausgestattet gegenüber den teureren CPUs. Die Größe dieses Caches lässt sich auslesen, da sie in den CPU-Kenndaten enthalten ist. Diese Kenndaten werden auch durch den Kernel ausgelesen und in der Datei "*/proc/cpuinfo*" hinterlegt. Dadurch besteht neben dem praktischen Test mit dem Programm **cachebench** (zumindest bei diesen bestimmten Prozessoren der Familie sechs) die Möglichkeit, zu prüfen, ob es es sich bei dem Prozessor um die preislich günstigere oder teurere Version handelt. Entsprechend wird bei diesen Prozessoren der aus der Datei "*/proc/cpuinfo*" ausgelesene Wert mit ausgegeben.

[5] auch mit i686 bezeichnet, dies trifft für die Prozessoren Pentium 2 und 3 der Firma Intel und Athlon (XP), Duron der Firma AMD zu

In der Datei "*cachebench.soll*" sind die Sollwerte für die einzelnen Cache–Ebenen sortiert nach CPU–Hersteller, -Familie und -Modell aufgelistet. Diese Sollwerte werden mit der Taktfrequenz des Prozessors multipliziert und mit den durch **cachebench** ermittelten Datendurchsatz für diese Cache–Ebene verglichen.

Das Programm **cachebench** wird mit den Optionen
<-O2 -DREGISTER -DUSE_INT> übersetzt. Ausgeführt wird es mit den Parametern
<-m 24 -e 1 -x 4 -d 5 -tr> für Rechnersysteme mit CPU der Firma Intel und
<-m 24 -e 1 -x 4 -d 5 -tb> für alle anderen Rechnersysteme. Durch diese Optionen werden die Tests bis zu einer Testgröße von 2^{24} = 16 MByte ausgeführt. Innerhalb jeder Zweierpotenz werden Messungen für vier unterschiedliche Testgrößen ermittelt. Jede Messung wird nur einmal durchgeführt und ist auf eine Laufzeit von fünf Sekunden begrenzt. Alle ermittelten Datendurchsätze mit einer Testgröße von unter 2048 Bytes werden nicht beachtet, da diese zu große Sprünge vollführen. Der Algorithmus zur Ermittlung der Cachegrößen und -geschwindigkeiten könnte dazu veranlasst werden, die erste Stufe des 1st–Level–Caches innerhalb dieser 2048 Bytes zu ermitteln.

Um die Cache–Ebenen zu ermitteln werden immer drei aufeinander folgende Werte des durch **cachebench** ermittelten Datendurchsatzes betrachtet. Ist der zweite dieser Werte kleiner oder gleich 90 % des ersten und der dritte nicht größer als der zweite Wert, so gilt der erste Wert als Ende des Cache-Plateaus. Der zweite Wert und alle folgende, welche mehr als drei Prozent niedriger als deren Vorwerte sind, gehen nicht in die Berechnung des Datendurchsatzes für eine Cache–Ebene ein.

Zur Ermittlung des Datendurchsatzes des Hauptspeichers wird der Durchschnittswert der letzten beiden gemessenen Datendurchsätze berechnet. Ein Sollwert kann dafür jedoch nicht bestimmt werden. Die dafür notwendigen Informationen wie Art des Speichers, Latenzzeiten[6], Taktfrequenz des Frontside–Bus und des Datenbusses zwischen Northbridge und Hauptspeicher sind nicht abrufbar.

Falls auf dem Rechnersystem das Programm **gnuplot** vorhanden ist, wird aus den von **cachebench** errechneten Werten eine Kurve gezeichnet. Somit kann sich der Nutzer die von **cachebench** gelieferten Ergebnisse in Form eines Diagramms ansehen. Das Diagramm wird im Postscript–Format in der Datei "*cachegraph.ps*" abgelegt.

3.3.4 Qualip::memtest86

Dieses Skript untersucht die Konfigurationsdatei des Bootmanagers **lilo** ("*/etc/lilo.conf*"). In dieser Datei werden alle Bezeichnungen der Boot–Optionen (Schlüsselwort *label*) mit der Zeichenkette "memtest" verglichen. Wurde diese Zeichenkette gefunden wird mit der Anweisung
% lilo -R ["label-Name"]
der Bootmanager angewiesen nur für den nächsten Neustart des Rechners **memtest86** zu starten (unter der Annahme das **memtest86** durch diese Boot–Option gestartet wird). Es wird eine Warnung ausgegeben, fünf Minuten gewartet und anschließend durch den Befehl
% reboot
das System neu gestartet.

[6] abhängig vom Speichertyp, teilweise werden andere Latenzzeiten verwendet als die RAM–Bausteine durch darauf enthaltene SPD-EEPROMs vorgeben

Um ohne Einwirken des Nutzers den Speicher ausführlicher zu testen wurden im Quelltext von **memtest86** zwei Änderungen vorgenommen und es neu übersetzt. Nunmehr werden automatisch alle zwölf Tests ausgeführt statt der acht Standardtests. Das übersetzte Programm (**memtest.bin**) befindet sich im "*memtest86*"–Verzeichnis.

3.3.5 Qualip::nbench

Der Quelltext von **nbench** wird übersetzt und ausgeführt. Das Übersetzen des Programms erfordert eine Mathematik–Bibliothek ("libm.so"). Aus den Ausgaben des Benchmarks werden die drei auf ein Rechnersystem mit AMD K6–CPU (233 MHz) normierten Indices extrahiert. Diese Indices werden mit den Sollwerten verglichen. In der Sollwertdatei ist zu jedem Eintrag CPU–Hersteller, -Familie und -Typ, Memory-, Integer- und Floating–Point–Index angegeben. Die in der Sollwert–Datei angegebenen Indices werden mit der Taktfrequenz der CPU multipliziert und dann mit den Indices, welche **nbench** zurückliefert, verglichen.

3.3.6 Qualip::netperf

Es wird davon ausgegangen, dass die Routing-Tabelle (durch diese entscheidet ein Linux-Rechner, über welche Netzwerkkarte eine Anfrage an einen Fremdrechner weiterzuleiten ist) konfiguriert ist. Die zu testende Netzwerkkarte wird festgelegt, indem ein Server, welcher an diesem Test beteiligt ist, gewählt wird. Durch die Routing-Tabelle ist vorgegeben, über welche Netzwerkkarte dieser Server erreicht werden kann.

Alle Server, mit welchen ein Test durchzuführen ist, werden in eine Liste eingetragen. Für alle in der Liste eingetragenen Server werden die Tests mit insgesamt einer Minute Laufzeit durchgeführt. Dabei wird für jeweils 30 Sekunden der Datendurchsatz der Übertragung mittels TCP–Verbindung und UDP–Paketen getestet. Für den Test der TCP–Verbindung wird der Befehl
% netperf -H <Hostname> -l 30
ausgeführt. Der Datendurchsatz via UDP wird mittels
% netperf -H <Hostname> -l 30 -t UDP_STREAM – -m 30000
getestet. Der UDP–Test wird somit mit einer Größe der Pakete von 30000 Bytes ausgeführt.

In der Datei "*netperf.soll*" ist für jeden Server, zu dem der Datendurchsatz gemessen wird, jeweils der Soll–Wert für den Test via TCP und UDP festgelegt. Für alle Tests sind Sollwerte festgelegt, die auf einer Verbindung mit 100 MBit/s (vollduplex) basieren. So wurde für jeden zu testenden Datendurchsatz ein Sollwert von 91 MBit/s (UDP) und 89 MBit/s (TCP) festgelegt.

Die ermittelten Datendurchsätze in MBit/s werden aus den Ausgaben extrahiert und mit den Sollwerten verglichen. Bei dem Test mittels UDP–Paketen werden zwei ermittelte Datendurchsätze von **netperf** zurückgeliefert. Zum Vergleich wird der Datendurchsatz verwendet, welcher vom Server ermittelt wurde. Vom Server werden nur Antwortpakete gesendet. Da bei der Übermittlung von Daten mit Hilfe von UDP–Paketen nicht gesichert ist, dass diese auch ihr Ziel erreichen, wird nur die Datenrate der Antwortpakete beachtet.

Die Anzahl der fehlerhaften Frames[7] wird vor Beginn und nach dem Ende der Tests gemessen und verglichen. Diese Fehleranzahl sollte sich während der Tests nicht erhöhen.

3.3.7 Qualip::nfstest

Es wurde ein Skript entworfen, das unter Nutzung des Programms **bonnie++** den Datendurchsatz zu einem NFS–Laufwerk misst. Das Skript ruft **bonnie++** mit den Parametern <-f -n 0 -q> auf, liest vor und nach Ausführung von **bonnie++** die von der Netzwerkkarte zurückgelieferte Anzahl fehlerhafter Frames aus und gibt die Ergebnisse der Tests aus.

Für das Schreiben auf ein NFS-Laufwerk wurde ein höherer Sollwert festgelegt als für das Lesen. Beim Schreiben sendet der Client die Daten zum Server. Zum Lesen von Daten muss der Client eine Anforderung an den Server stellen und auf die Übertragung der Daten vom Server warten. Die Sollwerte basieren auf einer Verbindung mit 100 MBit/s (vollduplex) und sind auf 6500 KByte (schreiben) und 6000 KByte (lesen) festgelegt worden. Die Schreib- und Lese–Tests von **bonnie++** werden mit Dateien durchgeführt, die die doppelte Größe des Hauptspeichers aufweisen.

Es wurden Tests mit zwei Rechnern ausgeführt, deren Konfiguration in Tabelle 3.2 verzeichnet ist. Die beiden Rechner waren durch einen Switch Summit48 der Firma Ex-

Komponente	Server	Client
CPU	AMD Athlon 500 MHz	AMD Duron 750 MHz
Hauptspeicher	384 MByte	256 MByte
Festplatte	IBM DDYS-T36950N - SCSI	Western Digital WD300BB-32AUA1 - IDE
Netzwerkkarte	3Com 3c905B-TX	3Com 3c905B-TX

Tabelle 3.2: Komponenten der Rechner für die Tests zur Erstellung von *Qualip::nfstest*

treme Networks miteinander verbunden. Der Switch wies die Verbindungen zu beiden Rechnern als 100 MBit vollduplex aus. Auf dem Server war ein NFS–Kernel–Server eingerichtet (Kernel 2.4.18; NFS–Version 3).

Der **bonnie++**–Test lieferte auf dem Test–Client (Tabelle 3.2) folgende Resultate:

```
------Sequential Output------ --Sequential Input- --Random-
-Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
          7044   3  3906   4            9369   4 430.2   4
```

Dabei dauerte der Test 4 Minuten und 41 Sekunden. Der Testrechner war mit 256 MByte Hauptspeicher ausgestattet. Die Testdauer ist dabei linear zur Größe des Hauptspeichers. Dies erzeugt eine entsprechende Belastung des Netzwerkes.

Um die Dauer der Belastung des Netzwerks zu reduzieren wurden die Parameter, mit denen **bonnie++** aufgerufen wird um folgende erweitert <-r 64 -s 128>. Hierdurch wurde Größe der Testdateien auf 128 statt 496 MByte reduziert.

[7] durch die Netzwerkkarte ermittelte fehlerhafte Frames werden von dem Programm **ifconfig** ausgegeben, Summe über alle Netzwerkgeräte

```
------Sequential Output------ --Sequential Input- --Random-
-Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
          10947   5 10575   7           +++++ +++ 13966  66
```

Im Cache–Speicher von Linux befinden sich die in die Testdatei geschriebenen Daten. Bei einem Auslesen werden die Daten aus dem Cache gelesen und nicht vom Server. Wenn die Zeit für einen Test weniger als 500 ms beträgt, so zeigt **bonnie++** statt des Ergebnisses des Tests nur "+" an.

Es wurde ein eigener Testablauf geschrieben. Dieser liefert unabhängig von der Größe des Arbeitsspeichers etwa eine gleiche Datenrate. Dazu ist es auch nötig, dass bei Client und Server weder von noch auf die Festplatte geschrieben wird. Denn je nach vorhandener Größe des Puffer–Speichers für das Schreiben und Lesen von Dateien bzw. Cache–Strategie des Kernels können sonst unterschiedliche Datenraten erzielt werden. Dies ist für den Client realisierbar, indem auf das Gerät "*/dev/null*"[8] geschrieben bzw. von "*/dev/zero*"[9] gelesen wird. Jedoch kann auf dem Server kein Link nach "*/dev/null*" und "*/dev/zero*" innerhalb eines im NFS exportierten Verzeichnisses angelegt werden, da diese vom Client als die entsprechenden lokalen Geräte interpretiert werden. Das Freigeben der Geräte "*/dev/null*", "*/dev/zero*" oder eines gemounteten *tmpfs*[10]–Verzeichnisses ist nicht gelungen. Es war möglich eine Ramdisk als NFS–Laufwerk freizugeben.

Zur Realisierung dieses Testablauf unter Nutzung einer Ramdisk seitens des Servers ist das Skript "*nfstest.run2*" entworfen worden. Es muss die Ramdisk auf dem Server mit mindestens 600 MByte freien Speicherplatz (nach der Formatierung) als NFS–Laufwerk freigegeben sein. Außerdem muss das Auslagern von Daten auf Festplatte (swap) deaktiviert werden. Das Skript erwartet auf dem NFS–Laufwerk eine Datei "*test.read*", welche eine Größe von exakt 300 MByte aufweist. Es werden unter Nutzung des Programms **dd** 300 MByte gelesen und geschrieben. Die dabei jeweils vergangene Zeit wird mit Hilfe des Systemaufruf *"gettimeofday"* gemessen. Nach dem Schreiben erfolgt ein Aufruf des Programms **sync**, um das Betriebssystem zu zwingen die gepufferten Daten zu schreiben.

Die Belastung des Netzwerks beträgt, bei Einhaltung der Soll–Werte von 6000 bzw. 6500 KByte/s, weniger als 103 Sekunden.

3.3.8 Qualip::smartsuite

Es wird bei allen Festplatten im Rechnersystem versucht, die SMART–Fähigkeit zu aktivieren und die SMART–Werte anschließend auszulesen. Dazu wird das Programm **smartctrl** mit den Parametern <-e> bzw. <-c> aufgerufen und die Ausgabe des Programms ausgewertet.

Zusätzlich werden bei IDE–Festplatten die Ausgaben der einzelnen (nummerischen) SMART–Werte[11] der Festplatten in der Datei "*smartsuite_log.out*" gespeichert. Es kann hierdurch bei einem eventuellen Fehler ein diesem Fehler vorhergehendes Ansteigen eines Wertes verfolgt werden. Dieses Ansteigen kann leider nicht automatisch ausgewertet

[8] an dieses Gerät kann alles nicht mehr benötigte gesendet (geschrieben) werden, da es nicht gespeichert wird

[9] liefert beim Ausgeben der Datei einen Datenstrom, welcher nur aus Nullen besteht

[10] *tmpfs* ist ein Dateisystem, welches alle darin gespeicherten Dateien im virtuellen Arbeitsspeicher hält.

[11] durch Aufruf von **smartctl** mit dem Parameter <-v>

werden, da es sich dabei auch um die Temperatur oder die Laufzeit der Festplatte handeln kann.

3.3.9 Qualip::sw-installation

Das Programm untersucht die Vollständigkeit der Software–Installation. Dazu sucht es in der folgenden Reihenfolge nach Paketmanagern.

1. dpkg - Debian Paketmanagement
2. rpm - Red Hat Paketmanagement
3. Slackware - kein Paketmanagement, installierte Software ist in dem Verzeichnis "*/var/adm/packages/*" aufgelistet

Die Liste der Software, welche installiert sein soll, erhält das Programm entweder per Parameter oder sie wird aus der Datei "*software.soll*" extrahiert.

Für jede Distribution muss diese Liste eigens geschrieben werden, da die Distributoren teilweise unterschiedliche Namen verwenden und die Software auf eine unterschiedliche Pakete verteilen.

3.3.10 Qualip::stabilitätstest

Das Ziel bei diesem Test war eine Auslastung des Rechnersystems (der Komponenten CPU, Grafikkarte und I/O-Subsysteme) von möglichst 100 Prozent zu erzielen. Eine 100 prozentige Auslastung zu erzielen, ist auf Grund von zwei Gründen nahezu unmöglich:

1. Die Komponenten sind bei ihrer Arbeit auf andere Komponenten angewiesen. Das heißt, dass zum Beispiel die CPU bei einer Auslastung einer anderen Komponente immer auch mit beansprucht wird. Dies kann wiederum dazu führen, dass die Komponenten auf Daten, Anweisungen oder Ergebnisse von anderen Komponenten warten müssen und so nicht optimal ausgelastet werden.
2. Die Anzahl der Unterkomponenten bzw. deren Eigenschaften sind verschieden. So verfügen die Prozessoren der Grafikkarten über unterschiedliche Fähigkeiten zur Darstellung dreidimensionaler Oberflächen. Je nach Grafikkarte erzeugen somit unterschiedliche Tests die höchste Last.

Tests, die speziell die Grafikkarte belasteten (**x11perf**, **glxgears** und **viewperf**), führten am häufigsten zu Abstürzen.

Dabei stellte sich heraus, dass zum Beispiel Grafikkarten mit dem Chipsatz Nvidia Geforce 2 MX 400 (mit Kühlkörper aber ohne Lüfter) der Belastung über einen längeren Zeitraum (ab einer Stunde) nicht mehr standhielten und das System zum Stillstand brachten. Dies war bei Tests mit **viewperf** und **x11perf** der Fall, wobei die Abstürze bei einer höheren Zimmertemperatur von 27-28°C vermehrt beobachtet wurden. Auch kann die Stromversorgung der Grafikkarten ein Problem für diese Instabilitäten sein [c't_08/2000] [Site-16]. Laut AGP–Spezifikation darf eine Grafikkarte sechs Ampere der 3.3 V Stromversorgung[12] entnehmen. Einige Mainboards können diese Stromstärke jedoch nicht auf

[12] AGP-Grafikkarten werden durch 3.3 V -, 12 V - und teilweise auch 5 V - Leitungen versorgt

Dauer liefern, was zu einem Absturz des Rechnersystems führt. Um diese Probleme zu lösen, wurde AGP-Pro entwickelt [c't_23/1999].

Ausgehend von den bisherigen Erkenntnissen kommen die Programme **cpuburn**, **burnMMX**, **glxgears**, **x11perf** und **viewperf** zum Einsatz. Diese Programme werden im ersten Teil des Stabilitätstests einzeln, anschließend teilweise in Kombination mit Kopierbefehlen oder dem Skript "*stabilitätstest1.sh*" ausgeführt. Dieses Skript wurde vom Autor als "Disk and CPU memory test" bezeichnet. Durch das Skript werden Daten dekomprimiert, kopiert und verglichen (mit dem Programm **diff**). Diese Operationen werden parallel ausgeführt. Der DMA–Controller[13] und der Hauptspeicher sollen durch die zu transportierenden Daten belastet werden. Das Skript wurde angepasst (es wurden etwa zehn Zeichen geändert). Falls **viewperf** nicht übersetzt werden kann, so wird anstelle dessen **x11perf** ausgeführt.

Da auch das Netzteil zu den zu testenden Komponenten zählt, muss ein Abschnitt des Stabilitätstests auf den größtmöglichen Stromverbrauch optimiert sein. Jedoch wäre es nicht praktikabel, eine Tabelle aufzubauen, in welcher die Stromverbräuche der Komponenten verzeichnet wären, um somit die Tests zu wählen, welche die höchsten Stromverbräuche der jeweiligen Komponenten erzielen. Das Rechnersystem muss in vielfältiger Weise getestet werden. Hierbei kann davon ausgegangen werden, dass unter anderem auch eine Test–Konstellation abgearbeitet wird, in der die theoretische, mit den Tests maximal erreichbare Strombelastung des Netzteils nahezu erreicht wird.

Teil des Stabilitätstests ist die Prüfung, ob der CPU in ausreichendem Maße gekühlt wird oder die Leistung des Prozessors herabgesetzt wird ("Throtteling"). Dies wird mit Hilfe der Programme **nbench** und **cpuburn** untersucht.

1. Dem Prozessor wird durch den **sleep**–Befehl für 10 Minuten Zeit gegeben, um nach eventuellen vorherigen Belastungen abzukühlen.

2. Es wird ein **nbench**–Test ausgeführt um die Leistungsfähigkeit nach Abkühlung zu ermitteln.

3. Für die in *Qualip::stabilitätstest* verwendete Standardlaufzeit für Testabschnitte wird das Programm **cpuburn** ausgeführt. Somit sollte bei nicht hinreichender Kühlung das Throtteling (Verringern der Arbeitsgeschwindigkeit des CPUs, falls unterstützt) einsetzen.

4. Ein erneuter Test mit dem Programm **nbench** wird ausgeführt.

5. In einem Vergleich wird festgestellt, ob die in den nbench-Tests ermittelten Indices einen Unterschied von größer oder gleich acht Prozent aufweisen. Falls dies der Fall ist, wird eine entsprechende Fehlermeldung ausgegeben.

Während der Durchführung der diversen Tests wird alle 2 Minuten das Programm **lmsensors** aufgerufen und die ausgegebenen Temperaturen aufgezeichnet. Diese werden nach dem Ende des Stabilitätstests ausgewertet. Es werden die maximale und die minimale Temperatur ausgegeben.

[13] vorausgesetzt der Festplatten–Controller bzw. die Festplatte nutzen den DMA–Controller

In der Datei "*stabilitätstest.out*" werden neben der Temperatur auch die Uhrzeit und der aktuell ausgeführte Test aufgezeichnet. Im Falle eines Absturzes lässt sich somit nachvollziehen, bei welchem Test und damit bei welcher Belastung Instabilitäten auftreten.

Das Signal zum Abbruch des Stabilitätstests wird durch *Qualip::stabilitätstest* abgefangen. Die gestarteten Hintergrundprozesse müssen erst beendet werden, welche andernfalls weiterhin ausgeführt würden. Dadurch kann es zu Verzögerungen kommen, bevor sich das Programm beendet.

3.3.11 Qualip::viewperf

Für die Startdatei des Viewsets ist die C–Shell nötig. Da sich die Bash–Shell zum Standard unter Linux entwickelt hat, wurde dieses Skript entsprechend abgeändert, um unter der Bash–Shell startbar zu sein.

Die Ausgaben des Skripts werden durch ein weiteres Skript **vpost.pl** verarbeitet. Dieses errechnet aus den dargestellten Bildern pro Sekunde und der Wertigkeit des jeweiligen Tests gemäß der Gleichung aus Kapitel 2.4.2 die Punktwertung für den Test. Extrahiert werden aus den Ausgaben der beiden Skripte der Autor der OpenGL–Bibliotheken, das OpenGL–Ausgabegerät und die erreichte Punktzahl.

Wie in Tabelle 2.7, Seite 49, zu sehen, steigt bei Erhöhung der Taktfrequenz der CPU, die Anzahl der Bilder pro Sekunde bei den Tests um unterschiedliche Faktoren. Somit ist davon auszugehen, dass bei den verschiedenen Tests die langsamste Stufe der Rendering Pipeline zum einen Teil bei der CPU und zum anderen Teil bei der Grafikkarte zu finden ist.

Auf Grund dieser Voraussetzungen müssen die Sollwerte für jede Kombination aus CPU und Grafikkarte festgelegt werden. Es ist nicht möglich einen Faktor für die Skalierung[14] der Sollwerte festzulegen.

Die Sollwerte ergeben sich aus der Bezeichnung der Grafikkarte, welche aus der Datei "*/proc/pci*" ermittelt wird und der gerundeten Taktfrequenz der CPU. Die Taktfrequenz muss gerundet werden, da sie bei Rechnersystemen mit gleicher CPU schwankt (+/-1 % der spezifizierten Taktfrequenz). Es werden alle den ersten beiden folgende Ziffern der (aus der Datei "*/proc/cpuinfo*" gelesenen) Taktfrequenz gerundet. Zu jeder Kombination aus CPU und Grafikkarte, die getestet werden soll, muss eine Punktwertung in der Datei "*viewperf.soll*" verzeichnet sein. Wird kein Sollwert gefunden, so wird das Ergebnis des Tests ohne Wertung ausgegeben.

3.3.12 Startdateien der Qualip-Suite

Es existiert eine Datei "*start*" im "*qualip*"–Verzeichnis. Mit dieser lassen sich zu startende Tests und deren Reihenfolge wählen. Die Ausgaben lassen sich als Text oder grafisch unterstützt mittels des Programmpakets **dialog** anzeigen. Folgende Parameter bietet die **start**–Datei:

-a Es werden alle Tests jeweils einmal ausgeführt (alphabetische Reihenfolge)

[14] beispielsweise mit der Taktfrequenz der CPU

-f Nach diesem Parameter folgt die Angabe der Tests, welche ausgeführt werden sollen. Es ist auch möglich Tests mehrfach auszuführen, indem sie mehrfach angegeben werden. Die Reihenfolge wird beachtet. Ein Aufruf könnte beispielsweise folgende Form haben.
% ./start -f sensors nbench sensors

-i Hiermit wird die grafische Unterstützung durch das Programmpaket **dialog** gewählt. Es sind automatisch alle Tests ausgewählt. Zu jedem Test wird eine Fortschrittsanzeige ausgegeben. Da die Grundlage dieser Fortschrittsanzeige nur ein Schätzwert ist, kann diese Anzeige sehr ungenau[15] sein. Die bei der Auswahl der auszuführenden Programme angezeigten Beschreibungen sind in der Datei "*Kurzbeschreibungen*" enthalten. Auch die geschätzte Laufzeit des jeweiligen Programms wird in dieser Datei vermerkt.

Falls der Test memtest86 gewählt wurde, so wird dieser auf Grund des dafür nötigen Neustarts generell als letzter Test ausgeführt werden.

3.4 Systemanforderungen

Die Qualip–Suite sollte auf jedem Linux–System ausführbar sein. Jedoch sind einige Anforderungen bezüglich benötigter Software bzw. Dateien zu erfüllen.

"/proc/cpuinfo" Die aus dieser Datei ausgelesenen Werte werden in diversen Unterprogrammen der Qualip–Suite genutzt, um Sollwerte zu skalieren.

tar, gzip, make, C-Compiler Diese Programme sind notwendig, um die als Quellpakete gelieferten Programme zu entpacken und zu übersetzen. Es wird von den Unterprogrammen *Qualip::nbench*, *Qualip::cachebench*, *Qualip::stabilitätstest* und *Qualip::viewperf* genutzt.

uname, date Um die ermittelten Ergebnisse einem Rechner und Datum/Uhrzeit zuzuordnen werden diese Programme benötigt. Alle Unterprogramme, außer *Qualip::memtest* verwenden diese Programme.

sensors Dieses Programm ist im lm-sensors Paket enthalten und fragt die Hardware–Sensoren des Systems ab.

x11perf, glxgears Diese Programme werden vom Teilprogramm *Qualip::stabilitätstest* der Suite genutzt.

xrefresh Um zu überprüfen, ob auf den X-Server zugegriffen werden kann, verwenden die Unterprogramme *Qualip::viewperf* und *Qualip::stabilitätstest* dieses Programm.

[15] Die Abweichungen können bis zu mehreren 100 % betragen. Speziell bei Festplattentests ist noch nicht bekannt, wie viele der vorhandenen Festplatten getestet werden können (auf Grund von Schreibrechten und dem freien Speicherplatz).

ifconfig Die Unterprogramme *Qualip::nfstest* und *Qualip::netperf* benötigen dieses Programm, um eine eventuelle Erhöhung der fehlerhaften Frames zu prüfen.

sfdisk Dieses Programm wird von dem Unterprogramm *Qualip::smartsuite* genutzt.

free Um die Größe des Hauptspeichers zu ermitteln, wird dieses Programm von den Unterprogrammen *Qualip::bonnie*, *Qualip::cachebench* und *Qualip::stabilitätstest* der Suite genutzt.

fdisk Die Unterprogramme *Qualip::stabilitätstest* und *Qualip::bonnie* nutzen dieses Programm zur Ermittlung der Größe von Festplatten.

3.5 Unterschiedliche Hardware-Architekturen

Die Suite wird generell auf jedem Rechnersystem ausführbar sein, das die Systemanforderungen erfüllt. Jedoch sind auf Rechnern, deren Architektur nicht i386–kompatibel ist, zusätzliche Vorbereitungen nötig. Die Teilprogramme *Qualip::bonnie*, *Qualip::memtest86*, *Qualip::netperf* und *Qualip::nfstest* verwenden für i386–Architekturen vorkompilierte Programme. Diese müssen durch die entsprechenden, für die zum Einsatz kommende Architektur übersetzten, Programme ersetzt werden. Die Internet–Adresse der Programme kann dem Anhang D entnommen werden, falls der Quelltext benötigt wird.

3.6 Einfluss von Hintergrundprozessen

Auf dem zu testenden Computer dürfen keine weiteren Prozesse gestartet werden, außer denen, welche für die Ausführung der Testprogramme nötig sind. Die einzelnen Benchmarks der Qualip-Suite reagieren hierauf mit unterschiedlicher Empfindlichkeit. Beispielsweise kann während des Laufs von **nbench** nebenbei eine Konfigurationsdatei mit einem Editor wie vi geschrieben werden, ohne dass die Ergebnisse mehr als fünf Prozent [16] verfälscht werden. Andererseits ist dies bei einem Lauf von **cachebench** nicht zu empfehlen. Wenn mit **cachebench** der Speicherdurchsatz gemessen wird, kann es in der aus den Ergebnissen resultierenden Kurve zu schwer nachvollziehbaren Einbrüchen kommen. Somit können falsche Werte für Größe und Geschwindigkeit der Caches zurückgeliefert werden. Es wird demnach empfohlen, während des Ablaufs der Qualip–Suite das Rechnersystem nicht zu nutzen und (temporär) nicht benötigte Hintergrundprozesse zu beenden.

[16] vorausgesetzt der Computer hat einen CPU-Takt von 1 GHz oder mehr

Kapitel 4

Zusammenfassung und Ausblick

Inwieweit eine neu entwickelte Software genutzt wird, hängt von dem Verhältnis zwischen Aufwand[1] und Nutzen der Software ab.

Die Qualip–Suite bietet einen Test der Grundfunktionalitäten des PC's. Sie kann sowohl zum Test vor einer Übergabe des Rechners an den Nutzer als auch zur Kontrolle bzw. zur Eingrenzung eines eventuellen Problems genutzt werden. Im Stabilitätstest wurden die häufigsten Absturz–Ursachen, welche von der Hardware oder Treibern ausgehen, geprüft. Die Grundkomponenten CPU, Hauptspeicher, Grafikkarte und I/O-Subsystem werden auf die für ihre Spezifikationen zu erwartende Leistungsfähigkeit geprüft. Es sind Tests eingebunden worden, welche Hauptspeicher und Festplatten auf Fehler prüfen. Die Netzwerkverbindung wird mit Tests unterschiedlicher Komplexität untersucht.

Die Hardware heutiger Rechner erwies sich dabei als zu komplex und zu schnelllebig, als dass diese bis ins Detail ausgemessen werden könnte (siehe Kapitel 2.3.1). Wenn diese detaillierte Messung erreicht werden würde, wäre schon neue Hardware verfügbar, für die ein neuer Test entwickelt werden müsste. Es waren robuste Tests gefragt, welche Messwerte liefern, die auf gleichen Rechnersystemen möglichste wenig schwanken und auf unterschiedlicher Hardware entsprechend höhere bzw. niedrigere Messwerte liefern.

Mit Hilfe der Programmiersprache Perl und deren Regular Expressions wurde eine Sprache gewählt, welche das Starten von Programmen und das Auswerten von deren Ausgaben mit wenigen Zeilen Quelltext erlaubt (im Vergleich zu anderen Skript- oder Programmiersprachen).

Nutzer dieser Suite werden zusätzliche Tests integrieren wollen, da sie andere Schwerpunkte zu testender Funktionalitäten setzen. Die zusammengestellte Suite stellt nur eine Grundlage dar, welche für die jeweilige Verwendung zu erweitern bzw. abzuändern ist. Deshalb wurde eine Basis geschaffen, durch welche sich zusätzliche Tests in kürzerer Zeit und in ein vorgegebenes einheitliches Schema integrieren lassen. Die kürzere Entwicklungszeit resultiert neben der verwendeten Programmiersprache auch aus den durch die Module bereitgestellten Funktionen zur Ermittlung von Eigenschaften des zu testenden Systems.

Auf dem Weg zur Entwicklung der Suite wurden im ersten Kapitel die zu testenden Eigenschaften eines Rechnersystems festgelegt, die im Rechnersystem vorhandenen Subsysteme aufgeführt und deren Besonderheiten, Einzelkomponenten und Funktionalitäten erläutert. Dabei wurden

[1] Einarbeitungszeit, Kauf der Software, Umstrukturierungen, Anpassung von Abläufen, Pflege

- Testmöglichkeiten der Subsysteme eines Rechnersystems,
- Möglichkeiten der Fehlkonfiguration,
- die Praxisrelevanz von Tests,

berücksichtigt. Die Zahl der potentiellen Testprogramme verringerte sich deutlich. So konnte die in Kapitel zwei folgende Wahl der Testprogramme im voraus begrenzt werden. Kapitel zwei beschäftigte sich mit den Funktionalitäten, dem Aufbau, den Ausgaben und der programmiertechnischen Realisierung der Testprogramme. Auf Basis dieses Wissens wurde eine Entscheidung getroffen, ob ein Testprogramm in die Suite einfloß oder nicht. Die gewählten Programme genügen folgenden Anforderungen:

- Prüfung der in Kapitel eins beschriebenen Eigenschaften der Rechnersysteme
- Eignung durch besondere Eigenschaften gegenüber anderen Testprogrammen
- Reproduzierbarkeit der Ergebnisse
- Möglichkeit des Auslesen der Ergebnisse durch ein übergeordnetes Programm

Die Umsetzung der Suite wurde in Kapitel drei beschrieben. Dabei stand die Wahl einer geeigneten Programmiersprache im Vordergrund. Diese Entscheidung wurde zum Teil auch durch eine externe Studie beeinflusst. Das Kapitel drei beschreibt die Kriterien und Gründe für die Wahl der Soll–Werte. Eine Angabe der Systemanforderungen informiert den Nutzer über Software, die für die Nutzung der Qualip–Suite notwendig ist.

Der praktische Einsatz der Software zeigt am deutlichsten auf, welche Details noch verbessert werden können. Einige schon in die Suite eingeflossene Erkenntnisse sind im Anhang A vermerkt.

Im Hinblick auf neueste Entwicklungen der Firma Intel, welche durch die Hyperthreating–Technik Funktionalitäten von Multiprozessorsystemen auch in Standard–Systemen integrieren will, sollte die Suite im Hinblick auf Testmöglichkeiten dieser Rechnersysteme ergänzt werden.

Es ist eine CD-Version der Suite wünschenswert, welche einen Computer auch ohne installiertes Betriebssystem testen kann. Dabei können dann allerdings die Ergebnisse der Tests **nbench**, **bonnie++**, **viewperf**, **cachebench**, **netperf**, **nfstest** nicht auf das tatsächlich installierte Rechnersystem übertragen werden, da sie von den verwendeten Treibern und Bibliotheken abhängen. Die Nutzung einer CD–Linux–Variante als Grundlage wie Knoppix würde diesen Prozess vereinfachen. Auf dieser CD–Version der Suite könnte der Stabilitätstest abgeändert werden, so dass Kopier–Operationen von der CD auf die Festplatte implementiert werden, um das Netzteil stärker zu belasten. Auch würde das Ausschalten der nicht benötigter Prozesse auf der CD–Version möglich sein, um das Umschalten zwischen den Prozessen zu minimieren. Somit können die gestarteten Programme die Hardware besser auslasten, da sie mehr Zeit der CPU zugewiesen bekommen.

Eine eventuelle Erweiterung der Suite um erweiterte Fortschrittsanzeigen und Ausgaben, welche dann allerdings nur unter X–Windows lauffähig wären, ist wünschenswert. Als Basis bietet sich hier die Perl–GTK Bibliothek an.

Anhang A

Erkenntnisse der Anwendung der Suite

A.1 Test der CPU

Nach Fertigstellung der Qualip–Suite ergab sich folgender Vorfall: Eine Firma musste auf Grund von Instabilitäten einen Rechner wieder zur Reparatur zurücknehmen. Der Rechner zeigte ein verbesserte Stabilität nachdem der Rechner zurückgeliefert wurde. Erst zwei Wochen später wurde registriert, dass der CPU mit einer Taktfrequenz arbeitete, welche 25 % unter der vom Hersteller spezifizierten lag. Die Qualip–Suite hatte dies auf Grund einer ungeeignet gewählten Bestimmung des Sollwerts nicht registriert.

A.2 Kernel mit unterschiedlicher Optimierung

Der Kernel von Linux lässt sich für unterschiedliche Architekturen (auch bei i386–kompatiblen Rechnersystemen) optimieren. So wurden auf einem Testrechner mit einer CPU Pentium 4 der Firma Intel zwei unterschiedliche Kernel getestet. Beide Kernel wurden mit dem Compiler gcc 2.95 übersetzt. Einer der Kernel war für i386–kompatible CPU's optimiert, der andere für die CPU Pentium 4. Die Messergebnisse der Qualip–Suite waren jedoch, bis auf Unterschiede im Toleranzbereich von 2 bis 5 % (je nach Anwendung), gleich bei der Nutzung der beiden Kernel. Bei der Ausführung der Programme der Qualip–Suite werden Funktionen der C- und Mathematik–Bibliothek genutzt. Diese waren für Rechnersysteme mit der CPU 80386 optimiert.

A.3 libc6-Bibliothek mit unterschiedlicher Optimierung

Zum Vergleich wurde die *libc6*–Bibliothek mit dem **gcc**–Compiler der Version 3.2 auf einem Rechner mit einer CPU Intel Pentium 4 übersetzt. Die *libc6*–Bibliothek wurde mit den Standardoptimierungen für i386-kompatible Systeme übersetzt. In einem nächsten Schritt wurden bei der Übersetzung der Bibliothek die Parameter des Compilers um die Optionen <-cpu=pentium4 -march=pentium4 -msse2> erweitert. Die Tabelle A.1 zeigt die Unterschiede, welche die Programme **nbench** und **cachebench** ermittelten.

	i386-optimiert	Pentium 4-optimiert
nbench-Memory-Index	10.84	10.82
nbench-Integer-Index	7.26	7.32
nbench-Floating-Point-Index	15.27	15.49
cachebench-Datendurchsatz 1st-Level-Cache [MByte/s]	11407	11713
cachebench-Datendurchsatz 2nd-Level-Cache [MByte/s]	7047	7501

Tabelle A.1: Ergebnisse der Programme **nbench** und **cachebench** bei unterschiedlicher Optimierung der libc6-Bibliothek

Anhang B

Beschreibung der Module der Qualip-Suite

B.1 *“qualipformat.pm”*

B.1.1 ergeb()

Diese Funktion dient zur Ausgabe der Ergebnisse. Sie erwartet einen oder zwei Parameter. Der zweite Parameter beschreibt die Datei, aus der gelesen werden soll. Wird dieser Parameter nicht übergeben, so kommt automatisch die Datei “*x.ist*” zur Verwendung, wobei “x” durch den aktuellen Verzeichnisnamen ersetzt wird. In der Datei wird in allen Zeilen nach der der Zeichenkette gesucht, die im ersten Parameter an die Funktion übergeben wurde. Wird eine solche Zeile gefunden, so werden alle folgenden Zeilen ausgegeben. Die Funktion hat keinen Rückgabewert.

B.1.2 getextract()

Mit Hilfe dieser Funktion werden die Regular Expressions für den aktuellen Test ermittelt. Der Funktion kann als Parameter eine alternative Datei zu “*<aktueller Verzeichnisname>.extract*” übergeben werden, aus der die Regular Expressions übernommen werden. Alle Zeilen der Datei werden als Liste von der Funktion zurückgeliefert. Dabei werden Zeilen, welche mit dem Zeichen “#” beginnen nicht in die Liste übertragen.

B.1.3 getoutput()

Diese Funktion dient dem Einlesen der Zeilen der Datei “*<aktueller Verzeichnisname>.out*” in eine Liste, die von der Funktion zurückgegeben wird. Es kann der Funktion als Parameter ein alternativer Dateiname übergeben werden. Die Funktion führt vor dem Einlesen das Programm **sync** aus, um sicher zu gehen, dass alle Daten geschrieben wurden.

B.1.4 getsoll()

Mit dieser Funktion werden bestimmte Daten einer Datei eingelesen. Im ersten Parameter wird der Funktion eine Regular Expression übergeben. Mit dem optionalen zweiten Parameter kann ein Dateiname übergeben werden, welcher statt
"<*aktueller Verzeichnisname*>*.soll*" verwendet werden soll. Auf jede Zeile dieser Datei wird versucht, die Regular Expression anzuwenden. Dabei werden alle Werte, die durch die Regular Expression extrahiert werden, in eine Liste geschrieben. Diese Liste ist der Rückgabewert der Funktion.

B.1.5 oeffne()

Es wird mit Hilfe dieser Funktion eine Datei geöffnet, in welche die formatierten Ergebnisse mit Wertung geschrieben werden. Falls nicht durch einen Parameter ein anderer Dateiname vorgeben wird, so kommt die Datei mit dem Namen
"<*aktueller Verzeichnisname*>*.ist*" zur Verwendung.

B.1.6 outpreamble()

Diese Funktion schreibt den im ersten Parameter übergebenen Text in die Datei
"<*aktueller Verzeichnisname*>*.out*" oder in die Datei, deren Name im zweiten Parameter übergebenen wurde. Die Datei wird neu angelegt bzw. überschrieben, außer dem Dateinamen im zweiten Parameter wird ein ">" vorangestellt.

B.1.7 schliesse()

Die Funktion *schliesse()* schließt die Datei, deren Name als erster Parameter übergeben wurde oder welche mit *oeffne()* geöffnet wurde.

B.1.8 schreib()

Dieser Funktion können zwischen null und vier Parameter übergeben werden. Es wird in die durch *oeffne()* geöffnete Datei[1] mittels eines einheitlichen Formates geschrieben. Wird der Funktion kein Parameter übergeben, so wird der Datei eine Leerzeile angefügt. Der in dem ersten Parameter enthaltene Text wird in das Feld zur Beschreibung des Tests eingefügt. Der zweite und dritte Parameter enthält den String für den Ist–Wert und Soll–Wert. Im vierten Parameter wird das Feld zur Wertung des Ergebnisses festgelegt. Falls die Funktion mit genau drei Parametern aufgerufen wurde, so wird die Wertung durch Vergleich der Felder des Soll- und Ist–Wertes bestimmt. Ist der Soll–Wert kleiner oder gleich dem Ist-Wert, so erscheint im Wertung–Feld "OK", andernfalls "–Failed–". Wurde die Funktion mit zwei Parametern aufgerufen, bleiben die Felder für Soll–Wert und Wertung leer. Falls für Soll- oder Ist–Wert eine Zahl übergeben wurde, so werden diese nur bis zur zweiten Nachkommastelle ausgegeben.

[1] Falls die Datei noch nicht geöffnet war, so wird dies durch einen Aufruf von *oeffne()* vollzogen.

B.1.9 uebersch()

Diese Funktion erlaubt das Ändern der Überschrift des einheitlichen Ausgabeformats. Die als Parameter übergebene Zeichenkette ersetzt die voreingestellte Überschrift (“Ergebnisse der Tests”).

B.2 *“qualipfunc.pm”*

B.2.1 getcpuinfo()

Um Informationen über die CPU zu ermitteln, extrahiert diese Funktion die gesuchten Werte aus der Datei“*/proc/cpuinfo*”. Es wird eine Liste mit folgenden Inhalt von der Funktion zurückgeliefert.

- der Name des Herstellers
- die Nummer der “cpu family”-Eintrags
- die Nummer des “model”-Eintrags
- die Taktfrequenz der CPU
- die Größe des Cache-Speichers (es ist nicht ersichtlich, welche Cache–Ebene damit referenziert wird[2])

B.2.2 verzeichnis()

Diese Funktion ermittelt pro Festplatte die Partition, in deren Grundverzeichnis geschrieben werden kann. Von diesen Partitionen wird pro Festplatte jeweils die Partition mit dem meisten freien Speicherplatz ermittelt. Die Funktion liefert einen Hash[3] zurück, welcher wiederum Listen referenziert. Die Schlüsselstrings des Hash sind die Geräte, welche die Festplatten referenzieren (beispielsweise “hda” für eine IDE–Festplatte). In der Liste sind die Werte Größe der Festplatte, freier Speicherplatz einer Partition auf dieser Festplatte und Verzeichnis, in dem diese Partition eingehangen ist, gespeichert. Alle Größenangaben sind in KByte. Ein Beispiel für ein solches Hash hat folgende Form.

```
%devs = (
   hda  => [ 80000 , 8000 , "/mnt/IDE1/" ],
   sda  => [ 40000 , 5000 , "/mnt/SCSI1/" ],
);
```

B.2.3 ram()

Der Rückgabewert dieser Funktion ist die Größe des Arbeitsspeichers in MByte.

[2] Praktische Erfahrungen haben gezeigt, dass es sich dabei um die Größe des 2nd-Level-Cache handelt, wenn der Die der CPU zwei Cache–Ebenen beinhaltet. Ist nur der 1st-Level–Cache auf dem CPU-Die integriert, so ist der Wert die Größe des 1st–Level–Cache.

[3] spezielle Art von Array, bei dem Werte nicht über numerische Indizes, sondern über Zeichenketten ermittelt werden

B.2.4 befehl()

Diese Funktion dient zum Ausführen von externen Befehlen mit definierter Reaktion auf Fehler. Im ersten Parameter wird der Befehl übergeben. Im optionalen zweiten Parameter kann der Funktion vermittelt werden, was in einem Fehlerfall geschehen soll. Es existieren drei mögliche Zeichenketten, die als zweiter Parameter übergeben werden können.

print Die Fehlermeldung erfolgt in Form einer Ausgabe.

warn Die Fehlermeldung wird als Warnung auf dem Standardfehler–Ausgabestrom ausgegeben.

die Das Programm wird abgebrochen und die Fehlermeldung ausgegeben.

B.2.5 netzwfehler()

In der Funktion *netzwfehler* werden zwei Summen der empfangenen, gesendeten und der fehlerhaften Frames über alle aktiven Netzwerk–Geräte des Rechnersystems gebildet. Der Rückgabewert ist eine Liste mit 2 Zahlen. Die erste Zahl beinhaltet alle gesendeten und empfangenen Frames, die zweite alle fehlerhaften Frames.

Anhang C

Beispiele für Sollwert-Dateien der Qualip-Suite

C.1 Qualip::bonnie

```
Groessen 9500 18000 99999999999
9500            6
18000           9
99999999999    15
```

Unter “Groessen” werden die Geschwindigkeitsklassen für eine Einordnung der Festplatte angegeben. Ab der darauffolgenden Zeile stehen die tatsächlichen Sollwerte der jeweiligen Geschwindigkeitsklasse. Alle Werte sind in MByte angegeben.

C.2 Qualip::cachebench

```
AuthenticAMD61  6.3     2.2
AuthenticAMD64  6.3     3.8
GenuineIntel65  4.5     1.5
GenuineIntel151 4.2     3.4
```

Die erste Gruppe der Zeichen besteht aus dem Herstellername der CPU (vendor_id), CPU–Familie und -Modell. Die zwei Floating–Point–Werte sind die Sollwerte des Datendurchsatzes von 1st- und 2nd–Level–Cache bei einer Taktfrequenz von 1 MHz. Diese Werte müssen mit der Taktfrequenz der CPU multipliziert werden, um die tatsächlichen Sollwerte zu ermitteln.

C.3 Qualip::nbench

```
AuthenticAMD61  0.0051  0.0046  0.0095
AuthenticAMD64  0.0052  0.0048  0.0095
GenuineIntel65  0.0042  0.0041  0.0074
GenuineIntel151 0.0038  0.0029  0.0057
```

Ähnlich zu der Sollwert–Datei von *Qualip::cachebench* setzt sich die erste Gruppe von Zeichen wieder aus dem Namen des Herstellers der CPU, der CPU–Familie und -Modell zusammen. Die sich anschließenden drei Floating–Point–Werte entsprechen dem Memory-, Integer- und Floating–Point–Index bei 1 MHz Taktfrequenz. Dies sind die auf den Rechner mit AMD K6-CPU (233 MHz) normierten Indices, welche von **nbench** ausgegeben werden. Werden die Floating–Point–Werte mit der Taktfrequenz der CPU multipliziert, ergeben sich die Sollwerte.

C.4 Qualip::netperf

```
0t 88
0u 86
1t 88
1u 86
```

Für jede getestete Verbindung (die Zählung beginnt bei null) werden die Sollwerte in MBit/s angegeben. Die Buchstaben “t” und “u” stehen für den Test bei Nutzung des TCP- bzw. des UDP–Protokolls.

C.5 Qualip::nfstest

```
6000  6500
```

Für das Schreiben und Lesen mittels einer NFS–Verbindung werden die unterschiedlichen Soll–Werte in KByte/s angegeben. Der erste bezieht sich auf die Lesegeschwindigkeit, der zweite auf die Schreibgeschwindigkeit.

C.6 Qualip::viewperf

```
nVidia Corporation NV17 [GeForce4 MX440] (rev a3)  2400  45
```

Der erste Wert ist die durch das Programm **lspci** ermittelte Grafikkarte. Es folgt mit der Zahl 2400 die gerundete Taktfrequenz der CPU und mit der Zahl 45 der Sollwert für die zu erreichende Punktzahl dieses Rechnersystems.

Anhang D

Verwendete Testprogramme

- **smartsuite**: Version 2.1
  `http://sourceforge.net/projects/smartsuite/`
- **ide-smart**: Version 1.4
  `http://lightside.eresmas.com`
- **nbench**: Version 2.1
  `http://www.tux.org/~mayer/linux/bmark.html`
- **cachebench**: Version 1.1
  `http://icl.cs.utk.edu/projects/llcbench/cachebench.html`
- **bonnie++**: Version 1.02c
  `http://www.coker.com.au/bonnie++/`
- **SPEC CPU 2000**: Version 1.2
  `http://www.spec.org/osg/cpu2000/`
- **unixbench**: Version 4.1.0
  `http://www.tux.org/pub/tux/benchmarks/System/unixbench/`
- **stream**: Version 4.0
  `http://www.cs.virginia.edu/stream/`
- **AIM 9**: Version 1.1
  `http://www.caldera.com/developers/community/contrib/aim.html`
- **cpuburn**: Version 1.4
  `http://users.ev1.net/~redelm/`
- **netperf**: Version 2.2pl2
  `http://www.netperf.org/netperf/NetperfPage.html`
- **iozone**: Version 3.124
  `http://www.iozone.org/`

- **SPEC viewperf**: Version 6.1.2 und 7.0
  `http://www.spec.org/gpc/opc.static/opcview.htm`
- **SPEC glperf**: Version 3.1.2
  `http://www.spec.org/gpc/opc.static/glperf.htm`
- **x11perf**: Version 1.5
  `http://www.xfree.org/`
- **lm-sensors**: Version 2.6.5
  `http://secure.netroedge.com/~lm78/`
- **CPU Burn-in**: Version 1.00
  `http://users.bigpond.net.au/cpuburn/`
- **glxgears**: keine Angabe, in XFree Verion 4.2 enthalten
  `www.xfree.org`
- **memtest86**: Version 3.0
  `www.memtest86.com`
- **mii-diag**: Version 2.06
  `http://www.scyld.com/diag/`
- **Disk and CPU memory test**:
  `http://people.redhat.com/dledford/memtest.html`

Anhang E

Glossar

BIOS - BIOS ist die Abkürzung für “basic input/output system”. Es handelt sich um eine Software auf einem ROM–Speicherbaustein, die die Nutzung von Grafikkarte, Tastatur und Laufwerken ermöglicht.

Cache - Dies ist ein Pufferspeicher zwischen Komponenten eines Rechnersystems, die in unterschiedlicher Geschwindigkeit Daten lesen oder schreiben.

Cache-Line Eine Cache-Line ist ein Datenblock im Cache. Die Cache–Line–Size ist die minimal geschriebene Datenmenge, wenn Inhalte des CPU-Caches überschrieben werden. Sie beträgt bei den 2nd-Level-Caches der i386–kompatiblen Prozessoren zwischen 32 und 128 Byte.

Floating-Point - Es handelt sich um eine gebrochene Zahl, welche zur Speicherung in Mantisse und Exponent zerlegt wird.

Frame - Ethernet-Datenpaket

i386 - Dies ist die Kurzform für die CPU 80386 der Firma Intel. Es handelt sich um den ersten Prozessor dieser Firma mit 32-Bit breiten Daten- und Adress–Bus. Dieser erlaubte in einer speziellen Betriebsart (Protected Mode) die Ausführung mehrerer Programme gleichzeitig.

Integer - Ganzzahl ohne Nachkommastellen

Mainboard - Dies ist die Systemplatine eines Rechnersystems. Sie bietet Schnittstellen für CPU, Speicher und Erweiterungskarten.

OpenGL - 3D Graphik-Bibliothek, von Silicon Graphics entwickelt

OSI-Referenzmodell - Abkürzung für “Open System Interconnection”: ein ISO Standard für Kommunikations–Protokolle, definiert sieben Schichten. Zu verarbeitende Pakete durchlaufen die Schichten in auf- bzw. absteigender Reihenfolge.

“*proc*”-Dateisystem Dies ist ein Virtuelles Dateisystem, dessen Dateien vom Kernel erst bei Zugriff erzeugt werden. Die Dateien enthalten Informationen zum Status des Rechnersystems. Auf bestimmte Dateien ist auch schreibend zugreifbar. Diese können zur Konfiguration des Systems genutzt werden.

Skript - Das Skript ist eine Liste von Kommandos, die durch den entsprechenden Interpreter (steht für gewöhnlich in der ersten Zeile der Skripts) abgearbeitet werden kann.

TCP - Dies ist die Abkürzung für “Transmission Control Protocol”. TCP erlaubt den Aufbau einer Verbindung zwischen zwei Netzwerk–Geräten in einem IP–Netzwerk, es trägt Sorge, dass Pakete ihr Ziel erreichen und in selbiger Reihenfolge wie abgesandt weitergegeben werden.

UDP - UDP ist die Abkürzung für “User Datagram Protocol”, ein verbindungsloses Netzwerkprotokoll, das den Versand und Empfang von Datenpaketen über ein IP–Netzwerk erlaubt.

Literaturverzeichnis

[SITE-1] Komprimierung mit dem Huffman Algorithmus
http://www.educeth.ch/informatik/interaktiv/kompression/huffman.html

[SITE-2] Die write back Cachestrategie
http://www.wimsbios.com/HTML1/advanced.html

[SITE-3] Entrollen von Schleifen
http://www.cray.com/craydoc/manuals/004-2518-002/html-004-2518-002/z828730246dep.html

[SITE-4] Online-Dokumentation von gcc
http://gcc.gnu.org/onlinedocs/

[SITE-5] die SPEC CPU2000 Ergebnisse auf
http://www.spec.org/cgi-bin/osgresults?conf=cpu2000

[SITE-6] Benchmarking Howto von André D. Balsa
http://oss.sgi.com/LDP/HOWTO/Benchmarking-HOWTO.html

[SITE-7] Linux Benchmark Suite
http://lbs.sourceforge.net/

[Site-8] webopedia.com - Definition PC
http://www.webopedia.com/TERM/P/PC.html

[Site-9] Geschichte von Unix und Linux
http://www.albion.com/security/intro-2.html

[Site-10] Definition Linux
http://www.auburn.edu/helpdesk/glossary/linux.html

[Site-11] inetd Superserver
http://www.linux-praxis.de/linux2/inetd.html

[Site-12] Dhrystone Benchmark
http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?Dhrystone

[Site-13] Whetstone Benchmark
http://www.tux.org/~balsa/linux/benchmarking/articles/html/Article1d-3.html

[Site-14] Technical Committee T13
`http://www.t13.org/`

[Site-15] Technical Committee T10
`http://www.t10.org/`

[Site-16] Messung der Stromaufnahme von Grafikkarten
`http://www.3dconcept.ch/artikel/stromverbrauch/index.html`

[Site-17] Gefahr: IDE-Festplatten im Dauereinsatz
`http://www.tecchannel.de/hardware/964/index.html`

[Preche] Lutz Prechelt - An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program, 2000
`http://www.ipd.uka.de/EIR/otherwork/jccpprt_computer2000.pdf`

[FlTeVe] Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling - Numerical Recipes in Pascal: The Art of Scientific Computing, 1989

[KerRit] Brian W. Kernighan, Dennis M. Ritchie - Programmieren in C, 2.Auflage, Hanser Verlag

[WaChOr] Larry Wall, Tom Christiansen, Jon Orwant - Programmieren mit Perl, Deutsche Ausgabe der 3.Auflage, O'Reilly Verlag

[MölHai] Tomas Akenine-Möller, Eric Haines - Real-Time Rendering, A.K. Peters Ltd.

[c't_21/1999] Ernst Ahlers - Watt ist nicht VA, c't 21/1999, Heinz Heise Zeitschriften Verlag

[c't_23/1999] Christof Windeck - AGP-Salat, c't 23/1999, Heinz Heise Zeitschriften Verlag

[c't_03/2000] Andreas Stiller - Speicherschieber, c't 03/2000, Heinz Heise Zeitschriften Verlag

[c't_07/2000] Andreas Bleul - Heizt nicht, c't 07/2000, Heinz Heise Zeitschriften Verlag

[c't_08/2000] Manfred Bertuch - Meister in 3D, c't 08/2000, Heinz Heise Zeitschriften Verlag

[c't_24/2000] Andreas Stiller - Bei Lichte betrachtet, c't 24/2000, Heinz Heise Zeitschriften Verlag

[c't_13/2001] Jörg Wirtgen - Die Letzten ihrer Art, c't 13/2001, Heinz Heise Zeitschriften Verlag

[c't_06/2002] Jörg Wirtgen - Leise Alleskönner, c't 06/2002, Heinz Heise Zeitschriften Verlag

[c't_08/2002] Christof Windeck - DDR-Technik, c't 08/2002, Heinz Heise Zeitschriften Verlag

[c't_13/2002] Andreas Stiller - Body-Check, c't 13/2002, Heinz Heise Zeitschriften Verlag

[c't_15/2002] Manfred Bertuch - Aufklärung in 3D, c't 15/2002, Heinz Heise Zeitschriften Verlag

[iX_07/1998] Uli Betzler, Michael Ehrig - Warten auf den Bus,iX 07/1998, Heinz Heise Zeitschriften Verlag

Hinweis: Für alle URLs gilt Stand 01.12.2002.