Projektarbeit

Bewertung der Compute-Leistung von Workstations mit SPEC-CPU-Benchmarks

-Installation-

-Verfahrensanalyse-

-Testdurchführung-

-Auswertung und Interpretation-

Autor: Carsten Mund

TU Chemnitz-Zwickau, Studiengang Informatik
Gruppe 03 FIF 91
Matrikelnummer 8956

Bearbeitungszeitraum: 22.05.1995 - 21.11.1995

Betreuer: Dipl.-Math. M. Ehrig, Universitätsrechenzentrum
Dipl.-Inf. S. Graupner, Fakultät für Informatik
LS Betriebssysteme

Inhalt

1. Einführung

1.1 Wurzeln und Ziele der SPEC

SPEC, die Standard Performance Evaluation Corporation, ist ein Zusammenschluß namhafter Computerhersteller, der 1988 gegründet wurde. Dies geschah mit der Zielstellung, realitätsnahe Benchmarkprogramme zu entwickeln, die leicht auf alle Plattformen portiert werden können und einen fairen Vergleich der Leistungsfähigkeit von Computern zulassen. Mittlerweile gehören der SPEC neben fast allen großen Hardwareherstellern auch einige Anwender, Softwarehäuser, Verlage und Einrichtungen aus dem Wissenschaftsbereich (unter anderen auch die TU Chemnitz-Zwickau) als Voll- oder assoziierte Mitglieder an. Eine aktuelle Liste der Mitglieder (Stand September 95) findet sich in [3], Seite 220.
Die Entwicklungs- und Portierungsarbeit wird hauptsächlich von den Hardwareherstellern geleistet. Sie sind es dann auch, die die ermittelten Ergebnisse als wichtiges Argument in der Werbung für ihre Produkte verwenden. Die SPEC-Benchmarks sind für jeden Interessierten, der die in den Prospekten und Datenblättern der Hersteller genannten Werte auf seiner eigenen Technik in der Praxis nachvollziehen oder hinterfragen möchte, gegen die Gebühr von je 150 bis 600 Dollar erhältlich. Dabei werden die vollständigen Quellen für die Benchmarks und die SPEC-Tools bereitgestellt. Es ist also keineswegs so, daß die Hersteller ein Geheimnis daraus machen, wie sie die Leistung ihrer Produkte bewerten. Allerdings ist es nicht selten so, daß die Bedingungen, unter denen die Hersteller testen, von denen im praktischen Einsatz meilenweit entfernt sind. Nicht zuletzt deshalb hat sich die SPEC strenge Regeln für den Messung (SPEC Run Rules [7]) und die Veröffentlichung (SPEC Report Rules [8]) der Benchmarkergebnisse auferlegt, um zumindest die Vergleichbarkeit zu gewährleisten. In diesem Zusammenhang spielen die "unabhängigen" Mitglieder, wie eben die Verlage und die Universitäten, eine wichtige Rolle. Deren Aufgabe muß es sein, die Aktivitäten der Hersteller kritisch zu bewerten, um Schwachstellen und dunkle Punkte aufzudecken. Außerdem sollten sie es sein, die die Hersteller dazu drängen, die Benchmarks an die neuesten technischen Entwicklungen anzupassen, diese in die Leistungsbewertung einzubeziehen und, was besonders wichtig erscheint, mehr Komponenten im Zusammenhang zu betrachten.
Die Entwicklung der SPEC-Benchmarks begann im Jahre 1989 mit der Veröffentlichung des ersten Benchmarksystems mit 10 CPU-intensiven Benchmarks (4 Integer-Tests, 6 Fließkomma-Tests), wobei die Durchschnittswerte SPECmark89, SPECfp89, SPECint89 ermittelt wurden.
1992 wurden diese durch zwei neue Suites abgelöst, die jeweils verschiedene Klassen von Benchmarks für typische rechenintensive Fest- beziehungsweise Fließkommaanwendungen enthielten. Neben reinen Laufzeitmessungen wurde angesichts der Entwicklungen bei Multiprozessormaschinen verstärktes Augenmerk auf die Ermittlung von Durchsatzleistungen gelegt. Die Ergebnisse SPECfp92, SPECint92, SPECrate_fp92, SPECrate_int92 stellen Relativleistungen zu einer Referenzmaschine (VAX 11/780) dar. 1995 werden diese beiden Suites durch eine neue Generation abgelöst, da moderne Rechner nur noch wenige Sekunden Laufzeit für einige der Benchmarks benötigen. Außerdem werden die Benchmarks an die typischen Eigenschaften heutiger Praxisanwendungen (speziell mehr Speicherbedarf) und an die aktuellen Entwicklungen im Technologiebereich (speziell an schnelle Cache-Speicher) angepaßt.
Neben diesen Paketen zur Bewertung der Prozessorleistung existiert seit 1992 die SDM-Suite (Software Development Multiuser), die Benchmarks aus einer Mischung von UNIX-Kommandos enthält, die multiuserbetriebstypisch sind. 1993 wurde der immer stärker fortschreitenden Vernetzung Rechnung getragen, indem der SPEC_SFS-Benchmark (System-Benchmark Fileserver) veröffentlicht wurde, mit dem die Leistung von Fileservern, die mit dem weitverbreiteten NFS arbeiten, bewertet werden kann.
Alle Benchmarks testen jedoch immer nur eine spezielle Komponente oder Eigenschaft moderner Rechnersysteme, was für spezielle Anwendungsgebiete mit differenzierten Anforderungen durchaus berechtigt ist. Ziel muß jedoch auch die Entwicklung eines Bewertungssystems sein, daß alle Komponenten (Netz, Prozessor, Speicher, Betriebssystem, Compiler) in ihrem Zusammenwirken betrachtet.
Die Testergebnisse der Hersteller werden im 4 mal jährlich erscheinenden SPEC-Newsletter veröffentlicht und sind mit Verzögerung im WWW unter [2] zugänglich.

1.2. Grundsätzliche Bewertungsverfahren für die CPU-Leistung

Die folgenden Aussagen gelten sowohl für die Integer-Benchmark-Suite CINT95, als auch für die Floatingpoint-Benchmark-Suite CFP95, deren Unterschiede in den Belastungsprofilen der einzelnen Benchmarks, jedoch nicht in den allgemeinen Prinzipien der Bewertungsverfahren liegen. Da sich die praktischem Messungen dieser Arbeit auf die CFP95-Suite beschränken, wird im folgenden bei der Nennung von Bezeichnungen hauptsächlich auf die CFP-Suite Bezug genommen, die äquivalenten Bezeichnungen für die CINT-Suite werden teilweise in Klammern angegeben.
Grundsätzlich muß zwischen 2 völlig verschiedenen Meßarten unterschieden werden, die bei flüchtiger Betrachtung leicht zur Irritationen führen können.
Bei der Ermittlung des Ergebniswertes SPECfp95 (bzw. SPECint95) wird die Laufzeit für jeden der 10 (bzw. 8) Einzelbenchmarks gemessen. Diese Zeiten werden im Anschluß in Relation zu den Laufzeiten auf der Referenzmaschine SPARCstation 10/40 betrachtet. Diese Relativleistungen (Referenzzeit SS10/40 in Sekunden / Gemessene Laufzeit in Sekunden) für die einzelnen Benchmarks werden als SPECratio bezeichnet und müssen für jede offizielle Messung einzeln veröffentlicht werden. Aus den 10 (8) SPECratio-Werten wird dann das geometrische Mittel berechnet, welches als Durchschnittswert SPECfp95 (SPECint95) für die relative Geschwindigkeit eines Rechners veröffentlicht wird. In Zeitschriften oder Datenblättern werden jedoch meist nur diese Durchschnittswerte genannt.

Geschwindigkeitskennzahlen für CFP95 :

$$\text{SPECratio (für 1xx.benchmark)} = \frac{\text{Laufzeit auf SPARCstation 10/40 (s)}}{\text{gemessene Laufzeit auf Testmaschine (s)}}$$

$$\text{Geom.Mittel SPECfp(\_base)95} = \left(\prod_{i=1}^{10} \text{SPECratio für Benchmark } i \right)^{1/10}$$

Für Multiprozessorsysteme ist dieser Relativgeschwindigkeitswert zum Vergleich nicht geeignet. Die Benchmarks selbst sind sequentielle Programme, die üblicherweise auf nur einem Prozessor ausgeführt werden. Inkorrekt ist es, die Einprozessorwerte einfach auf die Zahl der Prozessoren hochzurechnen. Selbst wenn die Benchmarks parallelisierbar wären, durch den Overhead für die Koordination und eventuelle Ressourcenengpässe (Speicher, Bus, Cache) kann ein solcher mathematische Idealwert nicht erreicht werden.
Von SPEC wurde daher ein spezielles Verfahren zur Messung der CPU-Leistung bei Multiprozessorsystemen defininiert, die "Homogenuous Capacity Method" - Durchsatz-Meßmethode SPECrate_fp95 (bzw. SPECrate_int95). Man ermittelt hierbei, wieviel gleichartige Jobs in einer bestimmten Zeit ausgeführt werden können. Entsprechend der Anzahl der Prozessoren werden mehrere Kopien eines Benchmarks gleichzeitig gestartet, es wird die Zeit vom Start der ersten bis zum Ende der letzten Kopie gemessen. SPEC erlaubt es auch, mehr Kopien zu starten, als Prozessoren vorhanden sind, was bei ausreichenden Ressourcen zu besseren Ergebnissen führen kann, da die Programme schneller auf den einmal im Instruction-Cache befindlichen gemeinsamen Programmcode zugreifen können.
SPECrate für die einzelnen Benchmarks ist dann definiert als die Anzahl benchmarkspezifisch normalisierter Jobs, die das System an einem Tag ausführen kann. Normalisiert bedeutet, daß die Ausführungszeit eines Einzelbenchmarks mit dem

Referenzfaktor = (Zeit d. Benchmarks auf SS10/40) / (Zeit d. längsten Benchmarks auf SS10/40)

multipliziert wird. Damit erhalten alle Benchmarks trotz unterschiedlicher Laufzeiten das gleiche Gewicht. Als vollständige Formel für die Einzelwerte SPECrate wird dann

$$\text{SPECrate (für 1xx.benchmark)} = \#\text{Kopien} * \frac{\text{Ref.zeit auf SS10/40}}{\text{längste Ref.zeit auf SS10/40}} * \frac{\text{Sekunden/Tag}}{\text{Laufzeit auf Testmaschine für \#Kopien}}$$

verwendet. Der Wert "Sekunden/Tag" ist 86400, die längste Einzellaufzeit auf SS10/40 ist 9600 Sekunden für 145.fpppp bei CFP95 (bzw. 4600 Sekunden für 099.go bei CINT95). Analog zu den Speedmessungen wird aus den Einzelwerten SPECrate der einzelnen Benchmarks wieder ein Gesamtdurchschnitt, diesmal für den Durchsatz, als geometrisches Mittel gebildet. Dieses ist der häufig veröffentlichte Wert SPECrate_fp95 (bzw. SPECrate_int95).

$$\text{Geom.Mittel SPECfp\_rate(\_base)95} = \left(\prod_{i=1}^{10} \text{SPECrate für Benchmark } i \right)^{1/10}$$

Wie für die Speedmessungen gilt jedoch auch hier, daß in den offiziellen Veröffentlichungen auch alle SPECrate-Einzelwerte genannt werden müssen.
Für Einprozessormaschinen ist die Veröffentlichung von Durchsatzergebnissen nicht vorgeschrieben. Diese kann jeder selbst aus den Laufzeiten berechnen. Es ist jedoch auch für Einprozessormaschinen erlaubt, Messungen mit mehreren Kopien durchzuführen und zu veröffentlichen. Die Benutzung eines gemeinsamen, im Instruction-Cache befindlichen Programmcodes bewirkt unter Umständen eine schnellere Abarbeitung bei mehreren Kopien. Umgekehrt kann die Abarbeitung durch den zusätzlichen Aufwand für Prozeßumschaltung und die erhöhten Ressourcenanforderungen auch verlangsamt werden. Diese architektur- und konfigurationsabhängigen Effekte sind zwar meßbar, i.a. jedoch relativ gering und nicht einzeln bewertbar, so daß üblicherweise nur "1-Kopie"-Ergebnisse für Einprozessormaschinen veröffentlicht werden.
Nach langer Diskussion wurde für die die neuen 95er Benchmarks ein weiteres Kriterium für die Differenzierung der Leistungswerte obligatorisch eingeführt, das von einigen Mitgliedern bereits in der letzten Phase der 92er Benchmarkgeneration als Standard gefordert und freiwillig benutzt worden war. Bis dahin war es möglich, jeden einzelnen Benchmark mit allen erdenklichen Optimierungseinstellungen zu übersetzen, solange nicht direkt an den Quellen geändert wurde, um die Abarbeitung zu beschleunigen. In der Praxis macht sich jedoch kein Anwender die Mühe, wegen ein paar Sekunden Zeitgewinn tagelang Compilerflags auszuprobieren, zumal dies erheblich zu Lasten der Übersetzungszeit geht. Die mit exotischen Optimierungen und zusätzlichen Präprozessoren geschönten Leistungsdaten spiegelten immer weniger die Alltagsleistungfähikeit wieder. Oftmals wurden auch "unsichere" Flags einbezogen, die gewissen Bedingungen und Prüfungen außerachtlassen und bei denen andere Programme oftmals abstürzen. Wegen dieser Auswüchse bei der Optimierung wurden mit den 95er Suites Baseline-Messungen zur Pflicht gemacht. Bei diesen darf nur ein einheitliches Set von Optimierungsschaltern für alle Benchmarks verwendet werden. Einzeloptimierte Resultate dürfen nur zusätzlich angegeben werden. Für den Anwender (Käufer) ergibt sich daraus eine bessere Überschaubarkeit und Vergleichbarkeit. Damit gibt es für die Floatingpoint-Suite (und äquivalent für die Integer-Suite) jetzt 4 statt bisher 2 die Rechenleistung wiedergebende Kennzahlen:

Basisoptimierte Kennzahlen: (obligatorisch)	SPECfp_base95	SPECfp_rate_base95
Einzeloptimierte Kennzahlen: ("aggressiv optimiert" - optional)	SPECfp95	SPECfp_rate95

Die unterschiedlichen Schemata bei der Namensbildung im Vergleich zur 92er Suite sollen wohl auch auf diese Situation hinweisen. (SPECfp_rate95 <--> SPECrate_fp92
SPECfp_rate_base95 <--> SPECrate_base_fp92)

An dieser Stelle sollen noch die Referenzwerte für die Auswertung der neuen CFP95- bzw. CINT95-Benchmarks genannt werden. Diese wurden Ende Juni 1995 von Bodo Parady, SMCC Performance Development gemessen und für diese Arbeit von Reinhold Weicker, SNI [1] zur Verfügung gestellt, obwohl die neuen Suites noch nicht verabschiedet waren, da Ablaufumgebung und Auswertungstools noch in der Entwicklung waren.

SPEC reference system

System:	SS10/40
Processor:	TI TMS390Z50 SuperSPARC
Clock:	40 MHz
L1 Cache:	16KB Data
	20KB Instruction
Other cache:	None
Operating System:	Solaris 2.4
Compilers:	SC3.0.1 13 Jul 1994
C compiler:	acc
Fortran:	f77
Baseline options:	-fast -xO4
Environment:	Multiuser(one user running)
Linker:	Default, dynamic

SPECfp95:

101.tomcatv:	3700
102.swim:	8600
103.su2cor:	1400
104.hydro2d:	2400
107.mgrid:	2500
110.applu:	2200
125.turb3d:	4100
141.apsi:	2100
145.fpppp:	9600
146.wave5:	3000

SPECint95:

099.go:	4600
124.m88ksim:	1900
126.gcc:	1700
129.compress:	1800
130.li:	1900
132.ijpeg:	2400
134.perl:	1900
147.vortex:	2700

1.3. Übergang von Testsuites CFP92/CINT92 zu Testsuites CFP95/CINT95 (Probleme bei Installation und Testdurchführung, sowie deren Einfluß auf den Gesamtverlauf der Projektarbeit)

Nach den allgemeinen Aussagen zu den Bewertungsverfahren ist es nun erforderlich, sich etwas mit dem derzeitigen Entwicklungsstand bzw. mit dem Stand während der Durchführung der Messungen zu dieser Arbeit auseinanderzusetzen.
Bekanntlich vollzieht sich in diesem Jahr der Übergang zu einer neuen Benchmarkgeneration. Motiviert wird dies durch die kurzen Laufzeiten der 92er Benchmarks auf modernen High-End-Maschinen. Außerdem müssen die einzelnen Benchmarkprogramme in ihren Anforderungen an das Verhalten heutiger und künftiger Applikationen angepaßt werden, um dem Anspruch realitätsnaher Leistungsmessung gerecht zu werden.
Bei Beginn dieser Arbeit am 22.Mai 1995 waren über die diese Problematik auch im URZ keine näheren Information vorhanden. Es standen die beiden Pakete CFP95.29 und CINT95.32 vom SPEC-ftp-Server zur Verfügung, von denen angenommen wurde, das es sich um offizielle, endgültige Versionen handelt. Mit diesen sollte diese Projektarbeit realisiert werden.
Bereits nach der Installation (Beschrieben in Anlage A) stellte sich heraus, daß erst ein Teil der Pakete aktualisiert und funktionsfähig war. So sind in diesen Paketen zwar neue 95er Benchmarkversionen enthalten, die gesamte Umgebung mit Menüsteuerung, Auswertung und Makefiles für diese Komponenten entstammte jedoch noch der 92er Suite und war völlig inkonsistent. Auch herstellerspezifische Files für die Übersetzung der Tools und der Benchmarks fehlen völlig. Die gesamte Dokumentation [5]-[8] gehört ebenfalls noch zur alten Version.
Anfragen an den SPEC-Verantwortlichen in Europa, Herrn Reinhold Weicker [1] blieben zunächst erfolglos, da dieser sich zu einem SPEC-Meeting längere Zeit in den USA aufhielt.
Es wurde begonnen, Messungen mit der Version CFP95.29 auf den verschiedenen Testrechnern durchzuführen, da die Benchmarks nach den Informationen aus den alten Dokumentationen auch ohne die Menüumgebung im Einzellauf gefahren werden können, was auch funktionierte.
Nach ca. einem Monat kam dann der erste Kontakt mit Herrn Weicker zustande. Dabei stellte sich heraus, daß es sich bei den Paketen um bereits überholte Zwischenversionen handelt. Allerdings war zu diesem Zeitpunkt auch noch keine endgültige Version verfügbar. Das Codefixing für die einzelnen Benchmarks war für den 27.6.95 angesetzt, die Entwicklung der Tools und der Umgebung sollte noch mindestens bis Ende August 95 dauern, da die gesamte Auswertung mit make, sed, awk auf PERL und gmake umgestellt wird.
Zu diesem Zeitpunkt stellte sich natürlich die Frage, wie mit dieser Projektarbeit weiter zu verfahren sei. Es standen 3 Alternativen zur Auswahl,
a) vollständiger Neubeginn mit der derzeit aktuellen, jedoch ebenfalls unfertigen 95er Version
b) vollständiger Neubeginn mit der noch gültigen, aber auslaufenden 92er Version
c) Fortsetzung der Arbeit mit der jetzigen Zwischenversion.
Es wurde daraufhin von allen Beteiligten die Entscheidung für Variante c) getroffen, da dadurch die meiste Kontinuität gegeben war. Sie wurde als die perspektivisch sinnvollere Variante gegenüber b) betrachtet. Der Wechsel zur aktuelleren, aber ebenfalls unfertigen Version hätte keine Vorteile gebracht, die bis dahin gemachten Messungen jedoch unnütz gemacht.
Gleichzeitig wurden die ursprünglichen Ziele dieser Arbeit modifiziert und konkretisiert. So wurde beschlossen, die Untersuchungen zunächst auf die Floatingpoint-Benchmarks zu beschränken. Auf die /uni/global-Installation wird verzichtet. (Es handelt sich um keine typische Kunden-Software, für die es ausreicht, ein paar Binaries in die entsprechenden Maschinenverzeichnisse einzuspielen. Die Benchmarks sollen auf den zu testenden Maschinen mit den dort verfügbaren Werkzeugen übersetzt werden. Dies im voraus für alle an der Uni vorhandenen Rechnertypen zu tun, dies vielleicht noch für verschiedene Optimierungsstufen, ist allein aus Speicherplatzgründen (die installierte Benchmarksuite inclusive Datenbasen benötigt ca. 50 MByte) abzulehnen, da diesem Aufwand kein entsprechend großer Nutzen gegenüber steht. Außerdem erscheint die Belastung der Uni-Rechner und des Netzes durch Nutzer, die "aus Spaß" stundenlang rechenzeitintensive Benchmarks fahren, nicht vertretbar. Für erforderliche Messungen zu Kauf- oder Aufrüstzwecken kann die Suite ja jederzeit individuell für spezielle Rechner oder Nutzer installiert werden.)
Eine Veröffentlichung der Benchmarkergebnisse im WorldWideWeb kann nicht erfolgen, da die Resultate mit einer inoffiziellen Vorversion ermittelt wurden. Außerdem sind unter Lastbedingungen ermittelten Ergebnisse nicht exakt reproduzierbar, die SPEC Reporting Rules [8] erlauben jeodch nur die Veröffentlichung von reproduzierbaren Testergebnissen.
Stattdessen ist das Augenmerk bei dieser Arbeit auf das Verfahren der Messung und die Interpretation zu legen.

In diesem Zusammenhang ist es nicht so interessant, ob sich für die einzelnen Benchmarks in der endgültigen Version noch Verschiebungen um ein paar Zehntelpunkte für die einzelnen SPECrate- und SPECratio-Werte gibt. Ziel ist eher die Untersuchung der Realitätsnähe von Meßmethode und Ergebnissen, sowie die Gewinnung von Informationen über den Einfluß verschiedener Randbedingungen. Auf abweichende Untersuchungsbedingungen und deren Folgen wird später noch vertieft eingegangen.

Das konkrete Vorgehen bei der Installation des Gesamtpaketes ist in der Anlage A dieser Arbeit beschrieben. Für die SPARCstation 20 Modell 612 (samson.hrz.tu-chemnitz.de) ist das vollständige Installationsprotokoll inklusive notwendiger Änderungen enthalten, bei den anderen Testrechnern wurde sich auf Ausschnitte mit Fehlern und deren Abhilfe beschränkt. Dabei wurde entsprechend der Beschreibung der Installation für CFP92 in der Datei README [6] im SPEC-Verzeichnis (./CFP95.29 nach Entpacken des Paketes CFP95.29.tar.Z) vorgegangen.
Die Erzeugung zusätzlich notwendiger Datenbasen für einzelne Benchmarks ist ebenfalls in der Anlage A , Teil II beschrieben. Mit Anwendung der Benchmarks selbst, sowie mit deren Installation, Übersetzung und Auswertung beschäftigt sich das folgende Kapitel 2.

2. Realisierung und Durchführung der Leistungsmessungen

2.1. Prinzipielle Realisierung

Die Benchmarksuite CFP95 (vorliegende Version CFP95.29) enthält insgesamt 10 rechenzeitintensive Einzelbenchmarks zum Test der Leistungsfähigkeit der Fließkommaarithmetik von Rechnern. Sie sind aus typischen numerischen Anwendungen abgeleitet, und zwar im einzelnen

101.tomcatv	-	Geometrische Transformationen (Netz-Generierung mit finiten Differenzen)
102.swim	-	Wettervorhersage (Lösung von "Wasser-Verflachungs-Gleichungen" unter Verwendung von Differenzen-Approximation)
103.su2cor	-	Quantenphysik (Berechnung von Massen von Elementarteilchen im Rahmen der Quark-Gluon-Theorie, Monte-Carlo-Simulation)
104.hydro2d	-	Astrophysik (Astrophysikalische Anwendung zur Berechnung von Raumschiffbahnen)
107.mgrid	-	Elektromagnetismus (Berechnung eines 3-dimensionalen Spannungsfeldes)
110.applu	-	Mathematik (Lösung eines Matrix-Systems über Pivots, Anwendung partieller Differentialgleichungen)
125.turb3d	-	Simulation (Simulation von Verwirbelungen in einem kubischen Raum)
141.apsi	-	Wettervorhersage (Berechnung von Statistiken über Temperaturen und Niederschläge, Matrizenbenutzung)
145.fpppp	-	Chemisches Anwendungsprogramm (Durchführung mehrfacher Elektronen-Ableitungen)
146.wave5	-	Elektromagnetismus (Lösung Maxwell'scher Gleichungen auf 2-dim. Netz)

Alle Benchmarks sind in FORTRAN unter Verwendung doppeltgenauer Gleitkommazahlen implementiert (außer 102.swim, dort Verwendung einfacher Genauigkeit). Die Kurzbeschreibungen entstammen der Dokumentation der CFP92-Suite [5][6] und [3]. Weitergehende Informationen über konkrete Algorithmen, allokierten Speicherplatz und speziell beeinflußte Ressourcen sind noch nicht verfügbar, die einzelnen Benchmarks werden nach und nach in den nächsten SPEC-Newsletters vorgestellt und näher analysiert.
Allgemein läßt sich sagen, daß auf relativ großen Datenmodellen (meist Matrizen) komplexe Operationen wie Ableitungen, Approximationen, Lösungen von Gleichungssystemen vorgenommen werden, die die Daten oftmals nicht in deren Speicherungsreihenfolge verwenden, um so Cache- und (wohl weniger gewollt) Page-Misses zu provozieren. Der Versuch einer tieferen Analyse von 2 ausgewählten Benchmarks wird in 2.3. gemacht.
Die Geschwindigkeitsmessung erfolgt mit einer sehr einfachen Methode nacheinander für jeden der Benchmarks oder für einzelne vom Nutzer ausgewählte. Nach einer Vorbereitung (Setzen von Suchpfaden, Umgebungsvariablen, Referenzfiles) wird das vorher compilierte Benchmarkprogramm in einer eigenen Shell gestartet. Für diese Shell wird die Laufzeit mittels des UNIX-Kommandos time gemessen. time liefert dabei 3 Werte für die Abarbeitung dieser Shell:

real -die Gesamtausführungszeit
user -die zugeteilte Prozessorzeit im User-Mode
sys -die im während der Ausführungszeit im System verbrachte Prozessorzeit ,

Diese werden in eine Datei time.out im jeweiligen Benchmarkunterverzeichnis ($SPECHOME/benchspec/$BENCHMARK) geschrieben. Dazu werden auch die Ausgaben und eventuelle Fehlermeldungen in Dateien protokolliert.
Anschließend, also nicht mehr in der Zeitmessung, wird verglichen, ob die berechneten Ergebnisse

($SPECHOME/benchspec/$BENCHMARK/result/clone_*/$BENCHMARK.in.out) mit den vorhandenen SPEC-Referenzwerten (in $SPECHOME/benchspec/$BENCHMARK/result.ref/$BENCHMARK.in.out) übereinstimmen, vorausgesetzt, der Benchmark terminierte nicht schon vorzeitig aufgrund eines Fehlers im Programmablauf. Mit awk wird der REAL-Wert des time-Kommandos aus der Protokolldatei time.out extrahiert und zusammen mit der Meldung "BENCHMARK COMPLETED SUCCESSFULLY" oder einer Liste der zu den Referenzwerten gefundenen Differenzen ausgegeben.
Die ausgegebene REAL-Laufzeit muß vom Nutzer in Relation zur Laufzeit des Benchmarks auf der Referenzmaschine SPARCstation 10/40 gesetzt werden und liefert so SPECratio für den einzelnen Benchmark auf der getesteten Maschine.
Die Durchsatzmessungen für SPECrate laufen prinzipiell nach der gleichen Methode ab. Entsprechend der vom Nutzer anzugebenden Anzahl n der zu startenden Kopien werden im Resultatsverzeichnis $SPECHOME/benchspec/$BENCHMARK/result n Unterverzeichnisse clone_0,...,clone_n-1 für die Ergebnisfiles $BENCHMARK.in.out und $BENCHMARK.in.err angelegt. (Bei der eben beschriebenen Geschwindigkeitsmessung mit einem Exemplar existiert dementsprechend nur das Verzeichnis $SPECHOME/...../results/clone_0. ---> default) Anschließend wird wieder eine Shell gestartet, deren Laufzeit mit time gemessen und im jeweiligen Benchmarkverzeichnis in das File time.out protokolliert wird. In dieser Shell werden die n gewünschten Kopien des Benchmarks gestartet. Die gemessene REAL-Zeit für die Shell ist somit die Laufzeit vom Start der ersten bis zum Ende der letzten Kopie, genau diese Zeit wird für die Berechnung von SPECrate (siehe Abschnitt 1.2.) benötigt. Die USER- und SYS-Zeiten für die gemessene Shell bilden jeweils die Summe der CPU-Zeiten im User- bzw. System-Mode über die n gelaufenen Benchmarkkopien.
Da die gesamte Messung mit einem oder zwei Kommandos ausgelöst wird und anschließend nutzertransparent abläuft, wird auf die Erläuterung dieses Vorganges im Detail und mit diversen Verzeichnis- und Dateinamen verzichtet. Auf einige wesentliche Details wird im Abschnitt 2.4. zur praktischen Durchführung der Messungen hingewiesen.
Zur genauen Analyse stehen die nachfolgend genannten Shellscripts bzw. Makefiles zur Verfügung, die bei einem vollständigen Durchlauf in dieser Reihenfolge zum Einsatz kommen, indem sie sich selbst nacheinander oder ineinander verzahnt aufrufen.

benchrun - ($SPECHOME/bin/benchrun)
- Hauptshellscript zum Start eines oder mehrerer Benchmarks, realisiert Parsing der Argumente und Aufruf des Make-Prozesses

Makefile - ($SPECHOME/benchspec/$BENCHMARK/Makefile)
- Compilierung der einzelnen Benchmarks
- Start des jeweiligen Meßvorganges
- Ausgabe der Gesamtlaufzeit nach erfolgreichem Benchmarkablauf

clear.sh - ($SPECHOME/benchspec/$BENCHMARK/clear.sh)
Aufgerufen vom Makefile,
- beseitigt die Ergebnisfiles der letzten Messung

setup.sh - ($SPECHOME/benchspec/$BENCHMARK/setup.sh)
Aufgerufen von Makefile,
- wertet Variable $SPECUSERS aus und erzeugt dementsprechend viele clone_*-Verzeichnisse für die mit $SPECUSERS spezifizierte Zahl von zu startenden Kopien
- prüft das Vorhandensein der notwendigen Eingabefiles $BENCHMARK.in und kopiert diese in alle clone_*-Verzeichnisse

benchmark.sh- ($SPECHOME/benchspec/$BENCHMARK/benchmark.sh)
Aufgerufen von einer Shell, die von Makefile gestartet wird und für die die Laufzeit mittels time gemessen wird
- startet rscript.sh für jedes clone_*- Verzeichnis

rscript.sh - ($SPECHOME/benchspec/$BENCHMARK/rscript.sh)
Aufgerufen von benchmark.sh (n-mal bei n Kopien)
- startet das eigentliche Benchmark-Executable und leitet die Ausgaben um in Ergebnisfile $BM.in.out und Fehlerfile $BM.in.err im entsprechenden Verzeichnis

$SPECHOME/....results/clone_* der jeweiligen Benchmarkkopie

compare.sh - ($SPECHOME/benchspec/$BENCHMARK/compare.sh)
Aufgerufen von Makefile
- prüft in jedem Kopie-Verzeichnis $SPECHOME/...results/clone_* auf das Vorhandensein von Corefiles, Fehlermeldungen und die Übereinstimmung der berechneten Ergebnisse mit dem entsprechenden SPEC-Referenz-File und versieht die gemessene Zeit bei erfolgreichem Durchlauf mit dem entsprechenden Prädikat

$SPECHOME stellt das Verzeichnis dar, in dem die Suite installiert ist. Nach dem Auspacken Paketes ist dies zunächst CFP95.29, was jedoch danach beliebig geändert werden kann. $BENCHMARK (oder $BM) stellt jeweils einen der Namen der 10 Einzelbenchmarks dar. clone_* stellt eines der bei $SPECUSERS=n erzeugten n Verzeichnisse clone_0,...,clone_n-1 dar, in dem jeweils eine der n Benchmarkkopien arbeitet.

2.2. Leistungsmessung und Auswertung unter Berücksichtigung von Randbedingungen

Die Benchmarksuite CFP95 wurde entwickelt, um die CPU-Leistung von Rechnern realitätsnah und fair messen und bewerten zu können. Dazu wurden 10 rechenzeitintensive Anwendungen ausgewählt und angepaßt, die speziell die Fließkommaarithmetik von Rechnern stark strapazieren.
Es wird in manchen SPEC-Veröffentlichungen betont, daß die Benchmarks (auch die SPECrate-Methode) die CPU-Leistung und nicht eine Systemleistung messen. Trotzdem darf man nicht den Fehler machen, die gemessenen und veröffentlichten Ergebnisse losgelöst von anderen, die Funktionsfähigkeit von Rechnersystemen maßgeblich beeinflussenden Komponenten und Bedingungen im Umfeld zu betrachten. Dazu gehören erstens die technischen Faktoren wie Hauptspeichergröße, Caches, externe Speicher, Netzanbindung usw.. Die zweite ebensowichtige Faktorengruppe bildet die Softwarebasis, hierbei spielen neben dem Betriebssystem besonders die Optimierungsfähigkeiten der verwendeten Compiler eine wichtige Rolle. Drittens muß man unbedingt die Umwelt eines Rechners in Betracht ziehen, hier kommt es auf die Einbindung in die übrige Rechnerwelt und die Belastung durch verschiedene Nutzungsformen an. Von einer fairen und gleichwertigen Beurteilung unter Berücksichtigung aller dieser Größen ist man allerdings noch weit entfernt.
Die Leistungsmessungen bei den Herstellern werden oft unter ziemlich realitätsfremden Bedingungen gemacht. So sind die Rechner in den Labors von der übrigen Rechnerwelt abgeschirmt. Die Benchmarks laufen immer als einzige Anwendung auf den zu testenden Rechnern und befinden sich direkt auf den lokalen Sekundärspeichern der zu testenden Maschine. Dazu herrschen sicherlich keine Engpässe bei Speicher- und Cachekapazität.
Der Käufer einer Maschine stellt vielleicht irgendwann fest, daß die in den Verkaufsprospekten der Hersteller genannten Leistungswerte in praktischen Anwendungen oft nicht annähernd erreicht werden. Mit den Benchmarkprogrammen wird dann versucht, die Ursache des Leistungsdefizits zu finden.
Ein anderes Herangehen an die Benchmarknutzung lieferte auch das Motiv für diese Projektarbeit. Es steht die Frage, ob es mit den SPEC-Benchmark-Suites möglich ist, bestehende Rechnersysteme im Praxisbetrieb zu untersuchen und aus den Meßergebnissen Erkenntnisse über notwendige Erweiterungen, Nachrüstungen, Konfigurationsänderungen oder auch für künftige Rechnerneukäufe zu gewinnen. Auch die Messung von Leistungswerten für vorhandene Maschinen, die beim Hersteller nicht mehr im Programm sind und folglich von diesen nicht mehr gemessen werden, ist für viele Anwender von Interesse, um die vorhandene Technik im kaum noch überschaubaren Markt einordnen zu können.
Die dritte Anwendungsrichtung für Benchmarks sind Untersuchungen, wie sich der Einsatz von zusätzlich entwickelter und installierter Software, die zeitweise oder permanent auf einer Maschine laufen soll (zum Beispiel Überwachungsmonitore, Backupsysteme, Ressourcen- oder Lizenzverwaltungssysteme), auf die Gesamtleistungsfähigkeit dieser Maschine auswirkt und wieviel Ressourcen von dieser speziellen Software verbraucht und folglich zur bestehenden Konfiguration hinzugefügt werden müssen.

Für alle genannten Herangehensweisen ist die reine Betrachtung der von der Herstellern ermittelten SPECfp95 und SPECrate_fp95 zur Ursachenforschung ungeeignet. Durch die erhebliche Abweichung der Meßbedingungen auf im praktischen Einsatz befindlichen Rechnern gegenüber den "sterilen" Laborbedingungen bei den Herstellern sind schlechtere Meßergebnisse vorprogrammiert. Man kann die Herstellerresultate natürlich reproduzieren, indem man die zur Dokumentation jeder Messung gehörenden Meßbedingungen nachstellt. Dies löst jedoch keinesfalls die Probleme der Bewertung der Leistung im täglichen, praktischen Einsatz. Kein Anwender wird die Zeit und das Geld aufbringen, die Ergebnisse der Hersteller nachzustellen.
Die recht primitiv anmutende Zeitmessung mit dem Unix-Kommando time liefert neben der gesamten Benchmarklaufzeit zwei weitere Werte, die zwar bei SPEC offiziell nirgendwo auftauchen, aber aufschlußreichere Informationen über die Beeinflussung der Messungen durch Randbedingungen liefern, wenn man sie unter Einbeziehung von Hintergrundinformationen über Konfiguration und die typische Anwendungslast interpretiert.
Das time-Kommando liefert 3 Zeitwerte für die Ausführungsdauer eines ihm übergebenen Programmes. In diesem Fall wird die Ausführungszeit einer Shell gemessen, in welcher der eigentliche Benchmark ein- oder mehrmals gestartet wird. Die ermittelten Werte sind

- die Gesamtlaufzeit als Differenz zwischen Start- und Endzeit
  (Diese Zeit wird als REAL-Zeit bezeichnet.)
- die Zeit, in der der Anwendung tatsächlich die CPU zugeteilt war (Nutzerprozeßzeit)
  (Diese Zeit wird als USER-Zeit bezeichnet.)

- die Zeit, in der die CPU für die Ausführung der Anwendung dem Betriebssystem zugeteilt war (Diese Zeit wird als SYS-Zeit bezeichnet.)

Im Rahmen der Benchmarkabarbeitung wird dem Nutzer nur die REAL-Zeit mitgeteilt, daraus werden dann SPECratio und SPECrate berechnet. Die anderen beiden Werte finden in den offiziellen Auswertungen keine Verwendung, stehen jedoch nach der Benchmarkausführung in den Files $SPECHOME/benchspec/$BENCHMARK/time.out in den einzelnen Benchmarkverzeichnissen zu Verfügung.

Sofort liegt der Versuch nahe, einfach die Summe aus USER- und SYS-Zeit für die Berechnung der Ratiowerte zu verwenden. Für die offizielle Veröffentlichung ist dies aber nicht zulässig.
In den wenigen schon vorhandenen Dokumentationsfiles findet sich der Hinweis, daß die Benchmarks wenig Ein- und Ausgabeoperationen vornehmen, und daß deshalb die Differenz zwischen REAL-Zeit und der Summe aus USER- und SYS-Zeit nur sehr gering ist. Für Laborbedingungen (zwar Multiusersystem, aber während der Messung Singleuser-Status) mag das zutreffen, in der Praxis wird die Differenz weniger durch die I/O-Operationen, als vielmehr durch die Anzahl von Nutzern und die damit verbundene Prozeßanzahl und Speicherbelastung bestimmt.
Die SYS-Zeit liegt im normalerweise auf sehr geringem Niveau (unter einer Sekunde), da die Benchmarks aus Portabilitätsgründen ohne System- und systemnahe Funktionen entwickelt wurden. Die späteren Messungen, z.B. bei HP-PA, werden das bestätigen.
In dieser Arbeit werden neben den SPECfp95 und SPECrate_fp95 zwei weitere Durchschnittswerte (und die entsprechenden Einzelwerte) in die Auswertung und Analyse einbezogen. Diese werden als US_fp95 und USrate_fp95 bezeichnet und sind als "Grenzwerte" für die tatsächliche CPU-Leistung zu betrachten. Berechnet werden die Werte analog zu SPECfp95 und SPECrate_fp95, nur wird statt der Gesamtlaufzeit (REAL-Zeit) die Summe aus USER- und SYS- Zeit als Berechnungsgrundlage verwendet.
US_fp95 und USrate_fp95 basieren damit auf der reinen Prozessorzeit ohne Wartezeiten auf Prozessorzuteilung und Ein- und Ausgaben, und stellen damit die "optimalen Grenzwerte" für die CPU-Leistung der Maschine dar, die in dieser Konfiguration erreichbar sind. Es ist noch die Definition eines weiteren Grenzwertes denkbar, der allein auf der USER-Zeit beruht. Hier bliebe auch die Speicherkonfiguration außerhalb der Bewertung, es würde tatsächlich nur die CPU-Leistung bewertet.
Große Unterschiede zwischen REAL-Werten und den Grenzwerten sind primär auf zwei Ursachen zurückzuführen. Zum einen werden hier die Messungen im echten Multiuserbetrieb gemacht (treffender ist eigentlich Multi-Processing), d.h. n Prozesse laufen auf einer CPU, wodurch sich die begrenzt zur Verfügung stehende Rechenzeit von diesen n Prozessen geteilt werden muß und sich die Gesamtausführungszeit bei n gleichmäßig CPU-intensiven Anwendungen im Extremfall ver-n-facht. Zum anderen befinden sich bei den hier vorherrschenden Meßbedingungen die ausführbaren Dateien, Ein- und Ausgabefiles nicht direkt auf dem zu testenden Rechner, sondern sind entfernt gespeichert. Dadurch verlängern sich unter Umständen die Ladezeiten und die (wenn auch von Haus aus geringen) I/O-Zeiten. Dieser Einflußfaktor ist jedoch geringer zu bewerten, da die modernen Netzmedien kaum schlechtere Übertragungsraten liefern als die Harddisks. Bei nicht zu extremer Netzlast dürfte durch die entfernte Ausführung also keine zu starke Verlangsamung der Benchmarkausführung hervorgerufen werden. Die Bewertung der Leistung des Filesystems ist mit der CFP95-Suite jedoch unmöglich, dazu muß man z.B. den SPEC-SFS -Benchmark heranziehen.
Ein weiterer interessanter Interpretationsgegenstand sind die reinen SYS-Zeiten. So läßt das Ansteigen dieser Zeiten darauf schließen, daß bestimmte Ressourcen nicht ausreichen und während der Abarbeitung vom Betriebssystem koordiniert werden müssen. Dies betrifft insbesondere die Ressource Hauptspeicher. Wenn dieser nicht ausreicht, werden Teile der Daten auf die externen Speichermedien ausgelagert. Dieses vom Betriebssystem organisierte und normalerweise völlig anwendertransparent ablaufende Paging läßt sich direkt an der Systemzeit ablesen, die bei Maschinen mit besonders wenig Speicher und hoher Nutzerfrequentierung bei besonders speicherintensiven Benchmarks in den Minuten- und im Extremfall in den Stundenbereich anwachsen kann. Durch die hinzukommende Wartezeit auf das Zurückladen steigt gleichzeitig die REAL-Zeit erheblich stärker an, als dies bei "nur" hoher Prozeßanzahl der Fall ist. Man kann das auch erkennen, indem man während der Benchmarkabarbeitung den Prozeßstatus beim Bechmarkprozeß beobachtet. Dies ist dann häufig der Wartezustand auf I/O-Operationen, obwohl die Benchmarks selbst kaum I/O-Operationen durchführen. Es wird also auf das Zurückladen ausgelagerter Speicherseiten in den Hauptspeicher gewartet.

Auf weitere Einzelheiten bei der erweiterten Leistungsbewertung mit den USER- und SYS-Zeiten und den Zusatzwerten US_fp95 und USrate_fp95 wird an den passenden Stellen in den Auswertungen der konkreten Messungen (Abschnitte 3.x.3) und in der Gesamtauswertung (Abschnitt 4) eingegangen.

2.3. Kurzanalyse von einzelnen Benchmarks

2.3.1. Benchmark 101.tomcatv

(Untersuchung Programmversion Version CFP95.29 - inoffizielle Pre-Release)

Anwendungsart: Meshgeneration - Erzeugung eines Gitter- bzw. Netzmodelles (Geometrische Transformation)

Verfahren: Beschreibung der Gleichungen und der endlichen Differenzen-Approximation in W.Gentzsch, K.Neves und H.Yoshihara: Computational Fluid Dynamics: Algorithms and Supercomputers. AGARDograph No. 311, August 1987

Gegenstand des zugrundeliegenden Praxis-Verfahrens:

Bei diesem Verfahren der finiten Differenzen nach Thompson wird von einem durch ein Netz aus Punkten und Verbindungen beschriebenen 2D- oder 3D-Modell eines realen Gegenstandes ausgegangen. Berechnet wird im Verfahren, wie sich eine Positionsveränderung an einem Punkt des Netzmodells in n Schritten (Iterationen) durch das Netzmodell fortpflanzt und wie weit diese Ausbreitung bei bestimmten Parametern (u.a. innerer Widerstand) läuft. Damit lassen sich z.B Verformungen von als Gitter modellierten Körpern simulieren.

Autoren: W.GENTZSCH FH REGENSBURG UND
GENIAS SOFTWARE GmbH
Erzgebirgstr. 2, 8402 Neutraubling, Germany

* angepaßt für PAR93 und SPEC von Bodo Parady ,November 93
* angepaßt für SPEC CFP95 von Reinhold Weicker, Oktober 1994

Die nachfolgenden Informationen wurden aus dem FORTRAN-Quelltext gewonnen, es werden keine Aussagen zum eigentlichen Algorithmus gemacht, der ein praktisches Verfahren realisiert, sondern es werden Informationen über die Realisierung im Sinne der Informatik, also Zugriffs- und Speicherreihenfolge und Arten der vorkommenden (komplexen) Operationen gewonnen.

Speicheraufwand:

Einzelvariablen und Konstanten:

8 Integervariablen (4Byte)	Index, Laufvariablen, Feldgrenzen	32 Byte
22 REAL*8 -Variablen (8Byte)	Gleitkommazahlen in DOUBLE PRECISION	176 Byte

Felder:

Insgesamt 7 Zwei- und 2 Eindimensionale Feldvariablen mit folgenden Bezeichnungen und Feldgrenzen. Alle Felder besitzen als Komponenten Gleitkommazahlen doppelter Genauigkeit (REAL*8 bzw. DOUBLE PRECISION). Die Felder werden in folgender Reihenfolge deklariert (und dementsprechend im Speicher angeordnet).

AA	(513,513)	513*513 (263169) Elemente a 8Byte	2105352 Byte
DD	(513,513)	513*513 (263169) Elemente a 8Byte	2105352 Byte
D	(513,513)	513*513 (263169) Elemente a 8Byte	2105352 Byte
X	(513,513)	513*513 (263169) Elemente a 8Byte	2105352 Byte
Y	(513,513)	513*513 (263169) Elemente a 8Byte	2105352 Byte
RX	(513,513)	513*513 (263169) Elemente a 8Byte	2105352 Byte
RY	(513,513)	513*513 (263169) Elemente a 8Byte	2105352 Byte

RXM (1000)	1000 Elemente a 8 Byte	8000 Byte
RYM (1000)	1000 Elemente a 8 Byte	8000 Byte

Dies ergibt einen Gesamtspeicherbedarf für die verwendeten Daten von ca. 14.1 Megabyte. Mit diesem großen Speicherbedarf wird eine zu starke Bevorteilung durch Cachespeicher vermieden.

Grobablauf:

- Einlesen der Initialisierungsdaten
  Füllen der Felder X(i,j) und Y(i,j) mit mit Daten aus dem File TOMCATV.MODEL, dies bedeutet Lesen von insgesamt 526388 REAL*8- Zahlen vom externen Speichermedium (Harddisk über Netz).

- Es existiert eine zentrale Iterationsschleife, die maximal 750 mal durchlaufen wird.
  Innerhalb dieser werden nacheinander unterschiedliche Manipulationen an den Matrizen vorgenommen. Dies geschieht in 5 doppelten und zwei einfachen Zählschleifen, wobei die Laufvariablen immer den gesamten Indexbereich der oben vereinbarten ein- und zwei-dimensionalen Felder abdecken. Die einfachen Zählschleifen werden bei jedem Iterationsschritt 511 mal durchlaufen, die doppelten Zählschleifen (ineinander geschachtelt) entsprechend 511*511 mal. Innerhalb dieser Schleifen werden bei jedem Durchlauf zwischen 2 und 17 Zuweisungen mit mehr oder weniger komplexen Verknüpfungen von Floatingpoint-Operanden in den rechten Seiten vollzogen. Man kann daher davon ausgehen, daß die Aufenthaltszeit in den einzelnen Programmteilen über die Gesamtlaufzeit betrachtet ziemlich gleichmäßig verteilt ist. Durch den relativ kurzen Programmcode ist dieser Benchmark trotzdem als relativ (befehls)cachefreundlich zu betrachten. Die in den Tests erzeugten auführbaren Progremme hatten Größen zwischen 22000 und 40000 Byte. Man kann von einer relativ hohen Hit-Rate im Instruction-Cache ausgehen, zumal meist etappenweise in kurzen Schleifenabschnitten operiert wird.

  Es werden alle vereinbarten Datenobjekte im Programm verwendet, für alle Felder gilt, daß dies auf rechten wie linken Seiten von Zuweisungen geschieht. Der erforderliche Hauptspeicherplatz von ca. 14,1 Megabyte muß also tatsächlich allokiert werden, es besteht keine Gefahr von etwaiger "Dead-Code-Elimination" durch den Compiler.

  Kern des Benchmarks sind die für numerischen Anwendungen typischen Matrizenoperationen. Dabei kann man von einem ausgewogenen Verhältnis verschiedener Zugriffsfolgen sprechen. Da sind unter anderem:
  - Spaltenweiser Zugriff auf Feldelemente, d.h. es wird nacheinander auf direkt aufeinanderfolgende Speicherstellen Bezug genommen, bzw. geschrieben. Dieser Fall ist als sehr (Daten-)cachefreundlich zu betrachten.
  - Zeilenweiser Zugriff auf Feldelemente, d.h. es wird auf nacheinander auf Speicherstellen Bezug genommen, die hier 513*8 Byte auseinander liegen, bzw. sogar um den doppelten Betrag, wenn in Zuweisungen gleichzeitig auf die vorherige und folgende Zeilenposition bzgl. des aktuellen Zeilenindex zurückgegriffen wird. Dies ist bereits als wesentlich ungünstiger für die Cachenutzung anzusehen, kann aber durch vorausschauende Cachestrategien noch ausgeglichen werden.
  - Verstreuter Zugriff auf verschiedene Speicherstellen, die unterschiedlich nah beieinander liegen. Hier ist von einer hohen Miss-Rate beim Caching auszugehen.

  Diese Zugriffe werden noch variiert, indem die Felder auch rückwärts durchlaufen werden. Weitere Variationen entstehen durch gleichzeitige Bezugnahme auf unterschiedliche Feldelemente gleicher Felder auf rechten und/oder linken Seiten von Zuweisungen, sowie durch Verwendung mehrerer, verschiedener Datenfelder in den Zuweisungen, wobei diese verschiedenen Felder im Speicher mehr oder weniger weit auseinander liegen. Der Benchmark fordert durch diese unterschiedlichen Zugriffsfolgen auch die Fähigkeiten des Compilers, eine möglichst effektive Umsetzung in die jeweils am besten geeigneten Adressierungsarten (indizierte Adressierung, basisrelative Adressierung, basisindizierte Adressierung usw.) vorzunehmen, da die Zeit für die Adreßberechnung für die Operanden bei dieser Menge von Zugriffen einen nicht unerheblichen Zeitfaktor darstellen kann.
  Durch die ausgewogenen Zugriffe ist dieser Benchmark bezüglich des Caching-Verhaltens trotz des insgesamt großen Speicherbedarfs als durchschnittlich zu betrachten, was auch dem Ziel dieser

anwendungsbasierten Benchmarks von SPEC entspricht.
Die rechten Seiten der Zuweisungen sind zusammengesetzte Fließkommaoperationen, die aus den 4 Standardoperationen + - * / sowie in einigen Fällen den eingebauten Funktionen ABS() und MAX() bestehen, die Operanden sind Feldelemente obiger Felder von Fließkommazahlen bzw. skalare Realwerte.
Matrizenoperationen sind i.A. sehr gut vektorisierbar, dies gilt auch für diesen Benchmark. Wie gut dies geschieht hängt von der speziellen Rechnerarchitektur und den Optimierungsfähigkeiten des zugehörigen Compilers ab.
Teile des Programms könnten auch parallelisiert werden, die normalen Standard-Compiler der einzelnen Systeme erzeugen jedoch i.a. keinen parallelen Code, zumindest nicht ohne spezielle Direktiven. Dies ist auch nur für Parallelrechnersysteme sinnvoll.

- Nach dem Ende der zentralen Iterationsschleife erfolgt noch die Ausgabe der berechneten Ergebnisse auf Sekundärspeicher (hier Harddisk über Netz). Dies sind 2*750 doppeltgenaue Floatingpoint-Werte (REAL*8 a 8Byte) sowie 750 Integerwerte.

Funktionen:

Es werden nur integrierte FORTRAN-Standardfunktionen verwendet. Dies sind zum einen die Ein- und Ausgabefunktionen READ und WRITE, sowie die mathematischen Funktionen MAX (Maximum zweier Zahlen) und ABS (Betrag einer Zahl). Im Benchmarkprogramm werden keine Funktionen definiert, folglich ist keine Lokalitäts- und Anforderungsuntersuchung für Funktionen erforderlich.

Einfluß von Ein- und Ausgaben:

Das Verhältnis Gesamtlaufzeit zur E-/A-Zeit ist groß, daß der Einfluß von Ein- und Ausgaben insgesamt gering ist . Versuche, bei denen nur die E-/A-Zeiten gemessen wurden, ergaben einen Anteil von ca. 1% für diese Operationen. Gleichzeitig konnte dabei ermittelt werden, daß bei Testläufen mit genügend Speicher, bei denen kein Paging erforderlich war, ca. 3/4 der gemessenen (und ohnehin geringen) SYS-Zeiten auf die Ein- und Ausgaben entfielen.

Beeinflußbarkeit der Benchmarklaufzeit:

Der Zeitaufwand wächst linear mit dem aus tomcatv.in eingelesenen Wert der Variablen ITACT (in dieser Version 750), mit der die Anzahl der Iterationsschritte für die Berechnung festgelegt wird. Der Speicheraufwand läßt sich mit der am Programmanfang vereinbarten Konstante NMAX steuern, da diese die Feldgrenzen für alle Felder bildet. Verändert man den Speicherbedarf, so geht damit zwangsläufig eine Laufzeitveränderung einher, schließlich muß man die zusätzlichen Datenelemente auch verarbeiten, damit sie nicht vom Compiler bei der sogenannten "DEAD-CODE-ELIMINATION" wieder beseitigt werden. Dazu muß man den aus tomcatv.in eingelesenen Werte für die Variable N, die die Anzahl der Schleifendurchläufe steuert, an die definierten Feldgrößen anpassen. Eine Verkleinerung des Speicherbedarfs ist schon allein mit dem Einlesen eines kleineren Wertes für N möglich.
Für größere Felder müßte man außerdem erst eine entsprechend größere Datenbasis zur Initialisierung erzeugen (TOMCATV.MODEL), sowie ein entsprechendes File für den Ergebnisvergleich.

2.3.2 Benchmark 102.swim

(Untersuchung Programmversion Version CFP95.29 - inoffizielle Vorversion)

Anwendungsart: Lösung von "Wasser-Verflachungs-Gleichungen" (Shallow Water Equations) durch endliche Differenzenapproximation (Wettervorhersage)

Verfahren: Basiert auf Artikel - THE DYNAMICS OF FINITE-DIFFERENCE MODELS OF THE SHALLOW-WATER EQUATIONS, BY ROBERT SADOURNY J. ATM. SCIENCES, VOL 32, NO 4, APRIL 1975.

CODE BY PAUL N. SWARZTRAUBER, NATIONAL CENTER FOR ATMOSPHERIC RESEARCH, BOULDER, CO, OCTOBER 1984.

* angepaßt für SPECpar-Suite von Bodo Parady
* angepaßt für SPEC CFP95 von Reinhold Weicker

Die nachfolgenden Informationen wurden aus dem FORTRAN-Quelltext gewonnen, es werden keine Aussagen zum eigentlichen Algorithmus gemacht, der ein praktisches Verfahren realisiert, sondern es werden Informationen über die Realisierung im Sinne der Informatik, also Zugriffs- und Speicherreihenfolge und Arten der vorkommenden (komplexen) Operationen gewonnen.

Speicheraufwand:

Einzelvariablen und Konstanten:

11 Integervariablen (4Byte)	44 Byte
17 Realvariablen (4Byte)	68 Byte

Felder:

Insgesamt 14 zweidimensionale Felder von Gleitkommazahlen einfacher Genauigkeit (real) mit Feldgrenzen N1 und N2. In dieser Version festgelegt auf N1=N2=513.

513*513 (263169) Elemente a 4 Byte --->	pro Feld 1 052 676 Byte
Für 14 derartige Felder ergibt sich ein Speicherbedarf (Daten) von	14 737 464 Byte

Dies ergibt einen Gesamtspeicherbedarf für die verwendeten Daten von ca. 14.1 Megabyte. Typische Cachegrößen werden damit bei weitem überschritten, so daß eine zu starke Laufzeitbeeinflussung durch Cachespeicher ausgeschlossen ist. Der Einfluß bleibt auf lokale Abschnitte während der Abarbeitung beschränkt. Bei Maschinen mit "normaler" Speicherkonfiguration (mind. 32 Mbyte) wird kein Paging erforderlich sein, außer es wird viel Speicher im Umfeld von anderen Applikationen verbraucht.

Grobablauf:

- Initialisierung
  Dies geschieht in diesem Benchmark durch Aufruf einer Prozedur (SUBROUTINE) INITIAL. Es erfolgt dabei keine Argument-Übergabe, die Prozedur arbeitet mit den globalen Variablen des Hauptprogrammes.
  Es werden 9 Werte (5 real, 4 integer) aus dem Initialisierungsfile swim.in gelesen, über die u.a. die Anzahl der späteren Iterationen gesteuert werden kann. Dieses Einlesen zur Laufzeit soll verhindern, daß bereits zur Compile-Zeit Berechnungen ausgeführt werden. Aus den eingelesenen Werten werden Initialialwerte für weitere Variablen berechnet, dazu gehören auch die Obergrenzen für die Laufvariablen der Zählschleifen.
  In 3 doppelten Zählschleifen (512*512 bzw. 513*513 mal durchlaufen) und 2 einfachen Zählschleifen

(512 Durchläufe) werden 7 der vereinbarten Datenfelder mit Werten gefüllt, wozu auch die Standardfunktionen ATAN( ), FLOAT( ), SIN( ), COS( ) verwendet werden, etwa die Hälfte der Operationen sind reine Kopieroperationen kompletter Speicherbereiche.

- Ausgabe von 3 Integer- und 4 Real-Werten mit zugehörigen Beschreibungstexten in das Ergebnisfile swim.in.out, diese dokumentieren die Parameter des Benchmarklaufes.

- Eintritt in die zentrale Iterationschleife, diese wird in dieser Programmversion 900 mal durchlaufen. In dieser zentralen Verarbeitungsschleife werden nacheinander 3 Berechnungsunterprogramme CALC1, CALC2 und CALC3 aufgerufen.

  SUBROUTINE CALC1:
  -Keine Argumente und lokale Variablen, es werden die globalen Datenobjekte des Hauptprogrammes verarbeitet. (---> kein zusätzlicher Speicherplatz)
  -Diese Prozedur enthält eine doppelte Berechnungsschleife mit 4 relativ komplexen Berechnungen und Zuweisungen. Diese beiden geschachtelten Zählschleifen werden insgesamt 512*512 mal durchlaufen. In den komplexen Berechnungen werden Feldelemente aus bis zu 4 Feldern miteinander verknüpft, dabei wird noch auf bis zu 4 verschiedene Elemente eines Feldes in einer Berechnung Bezug genommen, die im Speicher unterschiedlich weit verstreut liegen. Eine Bevorteilung durch Datencaches wird damit nahezu ausgeschlossen.
  In zwei weiteren einfachen Schleifen (je 512 Durchläufe) werden die eben berechneten Ergebnisse auf andere Speicherstellen im gleichen Feld kopiert, dabei werden einmal direkt benachbarte Speicherstellen gelesen, in der zweiten Schleife solche, die um einen konstanten Index im Adreßbereich verschoben sind. (---> beides günstig für Daten-Caching)
  Aufgrund der Durchlaufanzahlen und der Komplexität der Operationen wird innerhalb dieser Subroutine die meiste Zeit in der relativ kurzen, doppelt geschachtelten Berechnungsschleife verbracht, dies ist als sehr günstig für Instruction-Caches anzusehen.

  SUBROUTINE CALC2:
  -Keine Argumente und lokale Variablen, es werden die globalen Datenobjekte des Hauptprogrammes verarbeitet. (---> kein zusätzlicher Speicherplatz)
  In der Struktur gleicht diese Prozedur der eben betrachteten CALC1. Die doppelt geschachtelte Berechnungsschleife (512*512 mal) enthält nur 3 Zuweisungen mit Berechnungen, dafür wird darin auf bis zu 5 Felder und darin wiederum auf bis zu 4 Speicherstellen zugriffen.
  Bezüglich der Beeinflussung durch Caches und die Lokalität innerhalb des Codes gelten die Aussagen von CALC1.

  SUBROUTINE CALC3:
  -Keine Argumente und lokale Variablen, es werden die globalen Datenobjekte des Hauptprogrammes verarbeitet. (---> kein zusätzlicher Speicherplatz)
  Strukturell gleicht diese Prozedur den vorigen mit einer Doppelschleife (512*512 Läufe) und 2 einfachen Schleifen (je 512 Läufe). Speziell in der Doppelschleife ist die Komplexität wesentlich geringer als bei den anderen beiden Prozeduren. Hier werden nur Feldelemente aus 3 Feldern mit Konstanten multipliziert und anschließend addiert. Auf die Feldelemente wird dabei sequentiell in der Speicherreihenfolge zugegriffen, was den Aufwand für die Adreßberechnung gegenüber den beiden anderen Subroutinen erheblich reduziert. In den beiden einfachen Schleifen werden die berechneten Ergebnisse gespeichert und für den nächsten Iterationszyklus kopiert.

- Ausgabe von Ergebnissen zur Verifizierung der Korrektheit der Berechnung. Dazu wird die Zykluszanzahl (integer) und ein programmintern berechneter Zeitwert (real) ins Ergebnisfile swim.in.out geschrieben. In einer doppelten Zählschleife (512*512 Durchläufe) wird über die Werte von 3 Ergebnisfeldern jeweils eine Checksumme gebildet. Dabei werden die Felder wiederum in der Speicherfolge durchschritten. Diese 3 Werte (real) werden ebenfalls ins Ergebnisfile geschrieben. Die Korrektheitsprüfung verzichtet also auf die Ausgabe der gesamtem Ergebnisfelder, es werden nur Checksummen verglichen. (--->Minimierung des Aufwandes für "teure" Ausgaben)

Funktionen:

Das Benchmarkprogramm verwendet eine Reihe von FORTRAN-Standardfunktionen. Da sind die Ein- und Ausgabefunktionen READ und WRITE, sowie die mathematischen Funktionen ATAN( ), FLOAT( ), SIN( ), COS( ), MIN0( ) und ABS ( ).
Im Programm werden 4 Subroutinen (Prozeduren) definiert. Diese verwenden jedoch durchweg nur gemeinsame globale Variablen des Hauptprogrammes, es werden keine lokalen Datenobjekte definiert. Die Prozeduren dienen allein der Programmstrukturierung und verursachen keine zusätzlichen Ressourcenforderungen. Sie enthalten keine gegenseitigen Aufrufe oder gar Rekursionen. Durch die relativ gleiche Struktur der Unterprogramme ist anzunehmen, daß ihr Anteil an der Gesamtlaufzeit etwa gleich groß ist. Innerhalb der Prozeduren wird der größte Teil der Zeit in kurzen, sehr oft zu durchlaufenden Schleifen verbracht, dieser Benchmark ist daher als zumindest befehlscachefreundlich zu bezeichnen, da die Lokalität im Code ziemlich hoch und der Code auch nicht sehr groß ist. Die erzeugten ausführbaren Programme hatten bei den Tests Größen zwischen 24000 und 60000 Byte.

Einfluß von Ein- und Ausgaben:

Die Ein- und Ausgaben des Programmes beschränken sich auf wenige Real- und Integerwerte, sowie einige Zeichenketten. Der Anteil für E/A an der Gesamtlaufzeit des Programmes ist damit verschwindend gering.

Beeinflußbarkeit der Benchmarklaufzeit:

Der Zeitaufwand wächst linear mit dem aus swim.in eingelesenen Wert der Variablen ITMAX (in dieser Version 900), mit der die Anzahl der Iterationsschritte für die Berechnung festgelegt wird. Der Speicheraufwand läßt sich mit den am Programmanfang vereinbarten Konstanten N1 und N2 steuern, da diese die Feldgrenzen für alle 14 Real-Felder bilden. Mit Veränderung des Speicherbedarfs geht natürlich eine Laufzeitveränderung einher, schließlich müssen die zusätzlichen Datenelemente auch verarbeitet werden. Dazu muß man die aus swim.in eingelesenen Werte für die Variablen N und M, die die Anzahl der Schleifendurchläufe steuern, an die Feldgrößen anpassen.
Dies ist offensichtlich in der endgültigen Programmversion für SPEC-CFP95 gegenüber dieser Vorversion noch gemacht worden. In [3] findet sich auf S.8 bei einer Benchmark-Kurzbeschreibung bei 102.swim die Aussage, daß es sich um ein "Shallow Water Model" mit einem 1024*1024-Gitter handelt. Dies würde gegenüber dieser Version eine Vervierfachung des Speicherbedarfs auf 56,4 Mbyte bedeuten. Um diese Datenmenge in diesem Verfahren zu verarbeiten ist in etwa auch die vierfache Zeit erforderlich. Bei Betrachtung der Messungen auf HP Apollo Serie 700 Modell 735/125 (tantalus), für die offizielle Vergleichsergebnisse vorliegen, kann man Veränderungen an den Zeiten ablesen, allerdings nicht dieser Größenordnung der vierfachen Laufzeit. Es liegt die Vermutung nahe, daß dafür in der endgültigen Version die Anzahl der Iterationen gegenüber dieser Version reduziert wurde. Die genauen Änderungen sind erst bei Vorliegen der endgültigen Version ermittelbar.

2.4. Durchführung von Messungen

Installation:

Zunächst muß das vorliegende Paket CFP95.29.tar.Z ausgepackt werden. Anschließend werden die SPEC-Kommandos, SPEC-Auswertungstools und die SPEC-Manuals generiert und in die entsprechenden Verzeichnisse kopiert. Dies geschieht mit

```
make IDENT=<ident> bindir
```

vom SPEC-Wurzelverzeichnis aus. (Dieses wird in dieser Arbeit immer mit $SPECHOME bezeichnet.) <ident> ist der Indentifikator für den vorliegenden Maschinen-Typ (i.A. Herstellername z.B. hp). Abhängig davon werden beim Bau der SPEC-Tools spezielle Files M.<ident> in Ergänzung zu den Standard-Makefiles verwendet, die spezielle, plattformabhängige Einstellungen und notwendige Änderungen in den Quellen vornehmen. Derartige M.<ident>-Files sind in der vorliegenden Vorversion erst sehr vereinzelt enthalten. Daher ist es erforderlich, während der Installation selbst ein paar Änderungen vorzunehmen. Die Installation auf den 5 ausgewählten Systemen ist in der Anlage A beschrieben, dazu ist das vollständige Installationsprotokoll für samson.hrz (SOLARIS 2.3) mit den erhaltenen Fehlermeldungen und den daraus resultierenden Änderungen abgedruckt, bei den übrigen Systemen sind nur die Auszüge mit notwendigen Änderungen aufgeführt.

Start der Benchmarks:

Grundsätzlich stehen zwei Abarbeitungsmöglichkeiten zur Verfügung. Zum einen ist dies die menügeführte Konfiguration und Abarbeitung, bei der auch die Ratios und Rates mit berechnet werden. Diese Variante mit runfp ist in der vorliegenden Release noch nicht realisiert, das Paket enthält noch die komplette Script-Struktur der CFP92-Suite.
Also bleibt nur der individuelle Start der Benchmarks mittels des Kommandos benchrun und einer Argumentliste. Dies hat den Vorteil, daß die Benchmarks auch zeitgesteuert in Scripts gestartet werden können. Vor dem Beginn der Messungen müssen einige Umgebungsvariablen und Pfade gesetzt werden. Dies geschieht bei C-Shell-Nutzern mit

```
source cshrc                                                    .
```

Die allgemeine Kommandozeile für die Benchmarkabarbeitung hat folgende Struktur:

```
benchrun  target  ident  version  [benchmarklist]
```

target ist eine Anweisung an benchrun, welche Aktionen mit den Benchmarks ausgeführt werden sollen. Möglichkeiten sind:

compile	-Übersetzen der Benchmarks
run	-Start der Benchmarks, Übersetzung erfolgt, wenn keine Binaries vorhanden sind
compare	-Vergleich der BM-Ergebnisse des letzten BM-Laufes mit den SPEC-Referenzwerten
validate	-führt nacheinander compile, run, compare durch
clean	-beseitigt Ergebnisse und Hilfsfile des letzten Benchmarklaufes
clobber	-beseitigt Ergebnisse und die ausführbaren Benchmarkprogramme

Eine dieser Möglichkeiten muß angegeben werden!

ident ist wieder Identifikator für ein herstellerspezifisches Überlagerungsfile M.<ident> für das standardmäßige Makefile. In diesem können evtl. erforderliche, herstellerabhängige Änderungen und Ergänzungen vorgenommen werden. Meist dienen die M.<ident>-Files jedoch dazu, spezielle Compilerflags für die optimierende Übersetzung anzugeben. Außerdem können alternative Suchpfade für Compiler und Referenzfiles spezifiziert werden. Wie schon beim Bauen der Tools gilt auch hier, daß in der benutzten Pre-Release kaum solche herstellerspezifischen Files vorhanden sind. In einigen der Benchmarkverzeichnisse ($SPECHOME/benchspec/$BENCHMARK) finden sich jedoch Vorlagefiles M.generic, mit denen man selbst leicht Überlagerungsdateien erzeugen kann, was für die optimierten Messungen genutzt wurde.

Hier wird als Beispiel ein solches Überlagerungsfile M.samson gezeigt, bei benchrun müßte als ident hier folglich samson angegeben werden.)

```
samson% more M.samson
#
#  Generic Vendor Wrapper for SPEC benchmark
#  Mit den SCHALTERN der aehnlichen Maschine aus CFP92
#

#  BENCHNAME=tomcatv

MACHID=samson

F77=f77
#  FC=${F77}
FC=f77
OPT= -xO4
OBJOPT= -c
OBJ= o
EXTRA_FFLAGS= -fast -xO4
EXTRA_CFLAGS=
EXTRA_LDFLAGS=
EXTRA_LIBS=

TIME=/bin/time

STD_TGTS = validate all compile run \
            homogeneous concurrent time_concur h_compare \
            clean clear clobber compare save

$(STD_TGTS):
      date
      make \
      "MACHID=$(MACHID)"  \
      "F77=$(F77)" \
      "FC=$(FC)" \
      "TIME=$(TIME)" \
      "OPT=$(OPT)" \
      "OBJ=$(OBJ)" \
      "OBJOPT=$(OBJOPT)" \
      "EXTRA_LIBS=$(EXTRA_LIBS)" \
      "EXTRA_FFLAGS=$(EXTRA_FFLAGS)" \
      "EXTRA_CFLAGS=$(EXTRA_CFLAGS)" \
      "EXTRA_LDFLAGS=$(EXTRA_LDFLAGS)" \
      $@
samson%
```

Wenn kein entsprechendes M.<ident>-File vorhanden ist, wird das Standard-Makefile und dessen default-Parameter verwendet. Beim Aufruf mit benchrun muß jedoch unbedingt eine Bezeichnung ident angegeben werden, es kann dafür ein beliebiger Dummywert verwendet werden.

version kann ein beliebiger String (üblicherweise meist eine Zahl) sein, der zur Unterscheidung der abgespeicherten Protokolle verschiedener Benchmarkläufe dient. (Die Bildschirmausgaben werden jeweils zusätzlich in ein File protokolliert, um für spätere Auswertungen und Vergleiche zur Verfügung zu stehen. Die Abspeicherung der Protokolle erfolgt im jeweiligen Benchmarkverzeichnis $SPECHOME/tests.results/$BENCHMARK/ nach dem Dateinamensschema ident.version . Für das nachfolgende Beispiel bedeutet dies CFP95.29/tests.results/101.tomcatv/tant.1.base).
Auch für dieses Argument muß eine Bezeichnung (evtl. Dummywert) angegeben werden.

Die Angabe einer Liste [benchmarklist] der zu startenden Benchmarks ist nicht zwingend erforderlich. Ohne dieses Argument werden automatisch alle 10 Benchmarks der Suite ausgeführt, sonst nur die in der Liste genannten.
Von den genannten targets wurden nur clobber, compile und validate während der Tests benutzt. Compile wurde nur zum Test nach der Installation verwendet. Clobber wurde grundsätzlich vor jeder Messung angewendet, um sicherzustellen, daß bei anschließender Verwendung anderer Compilerflags auch wirklich alles neu übersetzt wird.
Die eigentlichen Messungen wurden grundsätzlich mit dem target validate gemacht. Eine solche Messung wird am folgenden Beispiel demonstriert:

```
[71] % source cshrc
...
[72] % benchrun validate tant 1.base 101.tomcatv
BENCHRUN:START 101.tomcatv, tant, 1.base Tue Aug  8 08:26:11 MESZ 1995
        date
Tue Aug  8 08:26:12 MESZ 1995
        make \
        "MACHID=tant"  \
        "F77=f77" "FC=f77" \
        "TIME=/bin/time" \
        "OPT=+O3" \
        "OBJ=o" \
        "OBJOPT=-c" \
        "EXTRA_LIBS=-lvec" \
        "EXTRA_CFLAGS=+O4 -Aa -D_HPUX_SOURCE +Onolimit +ESfic +Olibcalls" \
        "EXTRA_FFLAGS=+O3 " \
        "EXTRA_LDFLAGS=-Wl,-aarchive" \
        validate
        f77  +O3 +O3  -c tomcatv.f
f77: Multiple optimization levels specified; using level 3
tomcatv.f:
   MAIN:
        f77  +O3 -Wl,-aarchive -o tomcatv tomcatv.o  -lvec
Tue Aug  8 08:26:37 MESZ 1995
Running 101.tomcatv:
spiff -- scanned 1000 words from file #1
spiff -- scanned 2000 words from file #1
spiff -- scanned 1000 words from file #2
spiff -- scanned 2000 words from file #2
: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)
101.tomcatv: Elapsed Time: 1:02:39.3
Tue Aug  8 09:29:19 MESZ 1995
BENCHRUN:END 101.tomcatv, tant, 1.base, Tue Aug  8 08:26:11 MESZ 1995
[73] %
```

Die angegebene "Elapsed Time" umgerechnet in Sekunden (3759.3 s) relativ zur Laufzeit auf der Referenzmaschine (3700 s) ergibt für diese Messung also ein SPECratio = 1 .
Wie im vorigen Abschnitt erläutert, werden für weitere Auswertungen auch die USER- und SYS-Zeiten betrachtet, die vom time-Kommando geliefert werden. Diese Werte finden sich nach einem Benchmarklauf im File $SPECHOME/benchspec/$BENCHMARK/time.out .

```
[74]%more ./benchspec/101.tomcatv/time.out
real  1:02:39.3
user    12:16.2
sys         1.0
: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)
[75]%
```

In den meisten Fällen wurden die Benchmarks alle nacheinander ausgeführt und anschließend per Script der Inhalt der 10 verschiedenen time.out-Files zusammengetragen. Das obige Ablaufprotokoll ist, wie bereits erwähnt, zusätzlich als Datei verfügbar, in diesem Fall als Datei tant.1.base im Protokoll-Verzeichnis $SPECHOME/tests.results/101.tomcatv .
Die Dokumentation zur CFP92-Suite enthält den Hinweis, daß benchrun für Laufzeitmessungen, nicht aber für Durchsatzmessungen (SPECrate) geeignet ist. Dies muß hier richtiggestellt werden. benchrun selbst bietet keine Möglichkeit, die Anzahl zu startender Benchmarkkopien anzugeben. Im bereits beschriebenen Abarbeitungszyklus wertet das Shellscript setup.sh jedoch eine Variable SPECUSERS aus und leitet alle notwendigen Schritte (Verzeichniskopien usw.) aus dem Wert dieser Variable ab. Der Defaultwert für SPECUSERS ist 1, man kann den Wert jedoch einfach ändern, indem man vor dem Aufruf von benchrun eine Umgebungsvariable SPECUSERS auf den gewünschten Wert setzt. Dies wird wieder am Beispiel demonstriert:

```
samson%setenv SPECUSERS 2
samson%benchrun validate samson 8.baserate 101.tomcatv
BENCHRUN:START 101.tomcatv, samson, 8.baserate Fri Sep 15 12:04:51 MET DST 1995
date
Fri Sep 15 12:04:53 MET DST 1995
make \
"MACHID=samson"  \
"F77=f77" \
"FC=f77" \
"TIME=/bin/time" \
"OPT=-xO4" \
"OBJ=o" \
"OBJOPT=-c" \
"EXTRA_LIBS=" \
"EXTRA_FFLAGS=-fast -xO4" \
"EXTRA_CFLAGS=" \
"EXTRA_LDFLAGS=" \
```

```
validate
Fri Sep 15 12:05:55 MET DST 1995
Running 101.tomcatv: Copies=2
spiff -- scanned 1000 words from file #1
spiff -- scanned 2000 words from file #1
spiff -- scanned 1000 words from file #2
spiff -- scanned 2000 words from file #2
spiff -- scanned 1000 words from file #1
spiff -- scanned 2000 words from file #1
spiff -- scanned 1000 words from file #2
spiff -- scanned 2000 words from file #2
SAMS: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)
101.tomcatv: Elapsed Time: 1:19:58.4
Fri Sep 15 13:25:58 MET DST 1995
BENCHRUN:END 101.tomcatv, samson, 8.baserate, Fri Sep 15 12:04:51 MET DST 1995
samson%
```

Im entsprechenden time.out-File erhält man die USER- und SYS-Zeiten als Summe für alle Benchmarkkopien (hier für 2 Kopien).

```
real  1:19:58.4
user    49:35.3
sys         7.5
SAMS: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)
```

Diese am Tage gemachte Messung lieferte erheblich schlechtere Werte als jene, die im nächsten Kapitel tabelliert wird und unter geringerer Last gemacht wurde. Theoretisch kann die Summe für USER und SYS hier doppelt so hoch wie die REAL-Zeit sein.

Im Kapitel 3 sind die Meßergebnisse für die Versuche auf den 5 vorgegebenen Systemen dokumentiert. Es wurden stets mehrere Messungen an verschiedenen Tagen gemacht, von denen die Beste aufgeführt wird. Erwartungsgemäß wurden die besten Ergebnisse während der vorlesungsfreien Zeit erreicht. Besondere Effekte und die speziellen Meßbedingungen werden näher kommentiert.
Abweichend von den SPEC-Regeln zur Benchmarkabarbeitung und Veröffentlichung der Resultate werden dabei die einzelnen Leistungswerte von ihrer Bedeutung folgendermaßen definiert:

SPECfp95_base, SPECrate_fp95_base sind Durchschnittsergebnisse einer Messung, die unter Verwendung laufzeitoptimierender Compilerflags realisiert wurde, wobei für alle Benchmarks die gleichen Schalter benutzt wurden. Dies deckt sich mit der SPEC-Definition von SPECfp_base95 und SPECfp_rate_base95 in den Run- und Report-Rules [7][8].
Dagegen sind SPECfp95 und SPECrate_fp95 in dieser Arbeit Ergebnisse von Messungen ohne jegliche Optimierungsschalter (nur -O), während SPECfp95 und SPECfp_rate95 in den SPEC-Regeln als Ergebnisse von Messungen mit "aggressiver" Einzeloptimierung für jeden Benchmark definiert sind. Solche Einzeloptimierungen wurden in dieser Arbeit nicht gemacht, da das Ziel nicht im Finden der besten Optimierungsflags bestand.
Dieser Zusammenhang gilt gleichermaßen für die inoffiziellen "US....-Grenzwerte" in dieser Projektarbeit. Insgesamt vergleichbar sind also nur die "_base"-Werte.

3. Meßergebnisse für fünf ausgewählte Rechner

3.1. SPARCstation AXIL - 311 Modell 5.1 (herkules.hrz.tu-chemnitz.de)

3.1.1. Gerätekonfiguration

Hardware:

Modell:	SPARCstation AXIL - 311 Modell 5.1 (vergleichbar SPARCstation10/40)
Hersteller:	Sun Microsystems, Inc.
CPU:	Superskalar SPARC 50 MHz
Anzahl CPU's:	1
FPU:	Integriert
Primärcache:	20 KB(I) + 16 KB(D) on chip
Sekundärcache:	1 MB(D+I) Supercache off chip
Hauptspeicher:	32 MB
Externe Speicher:	

Software:

Betriebssystem:	SunOS 4.1.3
Compiler:	Sun FORTRAN 2.0.1. Compiler Sun C 2.0.1. Compiler
Filesystem:	NFS
Systemstatus:	Multi-User (Tests auch unter echten Multi-User-Bedingungen gelaufen)

weitere Daten:

IP-Name:	herkules.hrz.tu-chemnitz.de	
IP-Nummer:	134.109.132.11	Ethernet
	134.109.2.14	FDDI
Standort:	Straße der Nationen	

Diese Daten stammen direkt aus dem Universitätsrechenzentrum beziehungsweise aus den Veröffentlichungen des URZ im WWW .
(Adresse: http://www.tu-chemnitz.de/home/czi/admin/hosts/herkules.html)

3.1.2. Resultate der Messungen

a) SPECfp95 :

Die folgenden Ergebnisse wurden entsprechend der SPEC-Regeln (SPEC-Run-Rules [7]) ermittelt. Es werden die tatsächlichen Laufzeiten der einzelnen Benchmarks betrachtet. Die Auswertung orientiert sich an den SPEC-Regeln zur Veröffentlichung offizieller Ergebnisse (SPEC-Reporting-Rules [8]).

BENCHMARK	SPEC-REFERENZ-ZEIT SEKUNDEN	LAUFZEIT SEKUNDEN	SPECratio	
101.tomcatv	3700	2665,5	1,4	
102.swim	8600	1700,5	5,1	
103.su2cor	1400	2059,9	0,7	
104.hydro2d	2400	5931,8	0,4	
107.mgrid	2500	3715,9	0,7	
110.applu	2200	41114,9	0,1	
125.turb3d	4100	1224,8	3,3	
141.apsi	2100	813,9	2,6	*
145.fpppp	9600	5300,6	1,8	
146.wave5	3000	4285,1	0,7	

Geometrisches Mittel (SPECfp95): 1,04* (Keine Compiler-Optimierung)

Zusätzlich werden nachfolgende Zeiten betrachtet, die bei jedem Testlauf quasi als Nebenprodukt ermittelt werden. In offiziellen SPEC-Veröffentlichungen sind diese Daten nicht enthalten, sie stellen also nur eine Interpretationshilfe für die speziellen Anwendungsbedingungen in diesem Fall dar. Es handelt sich dabei um die USER- und SYS-Zeiten, also die Zeit, in der ein Benchmark tatsächlich einer CPU zugeteilt war.

BENCHMARK	SPEC-REFERENZ-ZEIT	GEMESSENE ZEITEN STD MIN SEK		SUMME USER+SYS SEKUNDEN	USratio	
101.tomcatv	3700	USER	2084,3	2088,5	1,8	
		SYS	4,2			
102.swim	8600	USER	1586,4	1589,5	5,4	
		SYS	3,1			
103.su2cor	1400	USER	1409,0	1449,2	1,0	
		SYS	40,2			
104.hydro2d	2400	USER	5491,6	5498,4	0,4	
		SYS	6,8			
107.mgrid	2500	USER	3330,9	3337,4	0,7	
		SYS	6,5			
110.applu	2200	USER	2650,8	4443,2	0,5	
		SYS	1792,4			
125.turb3d	4100	USER	324,9	342,5	12,0	
		SYS	17,6			
141.apsi	2100	USER	809,0	810,0	2,6	*
		SYS	1,0			
145.fpppp	9600	USER	5076,7	5079,1	1,9	
		SYS	2,4			
146.wave5	3000	USER	1734,9	1901,5	1,6	
		SYS	166,6			

Geometrisches Mittel US_fp95: 1,63* (Keine Compiler-Optimierung)

SPECfp95/US_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Der Benchmark 141.apsi bricht vorzeitig ab. In der Fließkommaarithmetik tritt ein Fehler (Ungenauigkeit oder Unterlauf) auf, der nicht behoben werden kann. Bezüglich dieses Fehlers wurde bei Reinhold Weicker [1] als SPEC-Ansprechperson in Europa angefragt, woraufhin mitgeteilt wurde, daß das Problem bekannt sei und in der offiziellen Release behoben sein wird. Der vorzeitige Abbruch verfälscht den Wert für SPECratio leicht nach oben.

Flags: OPT = -O
EXTRA_FFLAGS = -O (-sun4)
(Der Schalter -sun4 wird automatisch bei jedem Compilerlauf mit verwendet, obwohl er nirgends direkt angegeben wird)

Bei den Compiler-Optimierten Meßreihen ergaben sich folgende Ergebnisse:

BENCHMARK	SPEC-REFERENZ-TIME SEKUNDEN	LAUFZEIT SEKUNDEN	SPECratio base	
101.tomcatv	3700	1977,3	1,9	
102.swim	8600	1560,3	5,5	
103.su2cor	1400	1707,4	0,8	
104.hydro2d	2400	5172,9	0,5	
107.mgrid	2500	3229,7	0,8	
110.applu	2200	34219,2	0,1	
125.turb3d	4100	574,1	7,1	
141.apsi	2100	813,9	2,6	*
145.fpppp	9600	5132,4	1,9	
146.wave5	3000	3880,9	0,8	

Geometrisches Mittel (SPECfp95_base): 1,25* (Baseline-optimiert)

Die Werte sind nur geringfügig besser als die der nichtoptimierten Messung. Die geringe Verbesserung wird nur bei Betrachtung der reinen Prozessorzeit (USER/SYS) deutlich, da die reinen Laufzeiten auch durch die unterschiedlichen Meßbedingungen differieren könnten. Eine Beseitigung des Fehlers bei 141.apsi mit speziellen Schaltern für die Fließkommaarithmetik (-cg.., -fieee, -fnonstd) war nicht möglich. Die Compiler-"Optimierung" beschränkt sich auf den Schalter -O4.

BENCHMARK	SPEC-REFERENZ-ZEIT	GEMESSENE ZEITEN STD MIN	SEK	SUMME USER+SYS SEKUNDEN	USratio base	
101.tomcatv	3700	USER	1908,2	1911,0	1,9	
		SYS	2,8			
102.swim	8600	USER	1507,8	1509,7	5,7	
		SYS	1,9			
103.su2cor	1400	USER	1306,7	1334,2	1,0	
		SYS	27,5			
104.hydro2d	2400	USER	5059,1	5061,4	0,5	
		SYS	2,3			
107.mgrid	2500	USER	3160,5	3162,6	0,8	
		SYS	2,1			
110.applu	2200	USER	2579,3	4020,6	0,5	
		SYS	1441,3			
125.turb3d	4100	USER	309,6	322,5	12,7	
		SYS	12,9			
141.apsi	2100	USER	781,0	781,7	2,7	*
		SYS	0,7			
145.fpppp	9600	USER	4964,8	4967,5	1,9	
		SYS	2,7			
146.wave5	3000	USER	1707,8	1837,2	1,6	
		SYS	129,4			

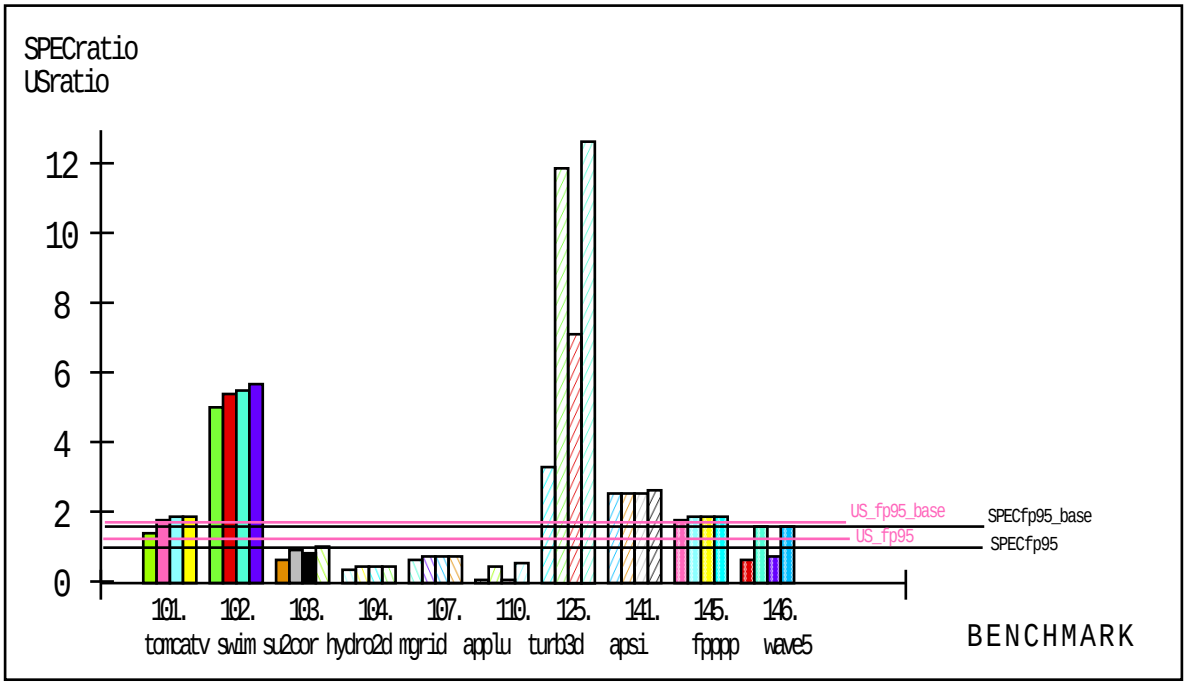
Geometrisches Mittel US_fp95_base: 1,72* (Baseline-optimiert)

SPECfp95/US_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Der schon erläuterte Fehler bei 141.apsi läßt sich auch mit verschiedenen in Frage kommenden Compiler-Flags nicht beseitigen .

Flags: OPT = -O
EXTRA_FFLAGS = -O4 (-sun4)

Im folgenden Diagramm werden für jeden Benchmark die 4 Meßwerte nochmals gegenübergestellt.



Die vier Säulen zeigen jeweils die Werte SPECratio,
USratio,
SPECratio(baseline_optimiert),
USratio(baseline_optimiert)

in eben dieser Reihenfolge für jeden der 10 Benchmarks.

Die vier Querlinien zeigen die Gesamtmittelwerte SPECfp95, SPECfp95_base und die zusätzlichen "Grenzwerte" US_fp95 und US_fp95_base.

b) SPECrate_fp95:

Da der Rechner Herkules nur eine CPU besitzt, ist keine spezielle Durchsatzmessung erforderlich, die Werte für SPECrate_fp95 ergeben sich durch Berechnung aus den Werten der Laufzeitmessungen in Teil a) :
(Für den ergänzenden "Grenzwert" USrate_fp95 ist an die Tabelle eine zusätzliche Spalte angefügt, die Werte sind jeweils aus der Summe aus USER- und SYS-Zeiten der Laufzeitmessungen berechnet, die hier nicht nochmals einzeln gelistet werden.)

BENCHMARK	REF-ZEIT SEKUNDEN	ANZAHL KOPIEN	ZEIT SEKUNDEN	SPECrate		USrate	
101.tomcatv	3700	1	2665,5	12,5		15,9	
102.swim	8600	1	1700,5	45,5		48,7	
103.su2cor	1400	1	2059,9	6,1		8,7	
104.hydro2d	2400	1	5931,8	3,6		3,9	
107.mgrid	2500	1	3715,9	6,1		6,7	
110.applu	2200	1	41114,9	0,5		4,5	
125.turb3d	4100	1	1224,8	30,1		107,7	
141.apsi	2100	1	813,9	* 23,2		23,3	*
145.fpppp	9600	1	5300,6	16,3		17,0	
146.wave5	3000	1	4285,1	6,3		14,2	

Geometrisches Mittel (SPECrate_fp95): 8,8*
(USrate_fp95): 14,7*

SPECrate_fp95/USrate_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Für den Fehler bei 141.apsi treffen die Bemerkungen aus 3.1.2. a) zu.

Flags: OPT = -O
EXTRA_FFLAGS = -O (-sun4)

Aus den Werten der optimierten Laufzeitmessungen ergeben sich folgende Durchsatzwerte:

BENCHMARK	REF-ZEIT SEKUNDEN	ANZAHL KOPIEN	ZEIT SEKUNDEN	SPECrate base		USrate base	
101.tomcatv	3700	1	1977,3	16,8		17,4	
102.swim	8600	1	1560,3	49,6		51,3	
103.su2cor	1400	1	1707,4	7,4		9,4	
104.hydro2d	2400	1	5172,9	4,2		4,3	
107.mgrid	2500	1	3229,7	7,0		7,1	
110.applu	2200	1	34219,2	0,6		4,9	
125.turb3d	4100	1	574,1	64,3		114,4	
141.apsi	2100	1	813,9	* 23,2	*	24,2	*
145.fpppp	9600	1	5132,4	16,8		17,4	
146.wave5	3000	1	3880,9	7,0		14,7	

Geometrisches Mittel (SPECrate_fp95_base): 10,6*
(USrate_fp95_base): 15,7*

SPECrate_fp95/USrate_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Für den Fehler bei 141.apsi treffen die Bemerkungen aus 3.1.2. a) zu.

Flags: OPT .= -O
EXTRA_FFLAGS = -O4 (-sun4)

Weitere Erklärungen finden sich in 3.1.3.

3.1.3 Erläuterung und Interpretation der HERKULES-Messungen

Eine Veröffentlichung offizieller SPECfp95- beziehungsweise SPECrate_fp95-Ergebnisse ist für dieses System nicht verfügbar. Ebenso sucht man vergeblich nach einem SPECfp92 -Ergebnis für diese Maschine. In etwa ist dieses Rechnersystem jedoch mit der SPEC-Referenz-Maschine, der SPARCstation 10/40, vergleichbar.
Der Rechner Herkules ist aufgrund der begrenzten Ressourcen gut für die Untersuchung der Einflüsse durch verschiedene Lastverhältnisse geeignet. Dies ist besonders auf die geringe Speicherbestückung mit "nur" 32 MByte zurückzuführen. Dennoch wird diese Maschine von vielen Nutzern, speziell aus den Fachbereichen Mathematik, ET, Maschinenbau verwendet, um dort ihre Anwendungen (Simulationen etc.) ablaufen zu lassen. Dies bringt die Maschine oft an den Rand ihrer Leistungsfähigkeit, was sich auch im normalen Betrieb an Antwort- und Abarbeitungszeiten feststellen läßt.
In diesem Zusammenhang muß erwähnt werden, daß die angegebenen Laufzeiten jeweils die besten mit den entsprechenden Optimierungsflags erreichten Werte sind. Bei der schlechtesten Messung (während der Vorlesungszeit, am Tage gestartet) wurde teilweise die 4-fache REAL-Zeit gemessen. Daraus resultieren entsprechende SPECratio- und SPECrate-Werte, die nur noch bei einem Viertel der hier dokumentierten Werte liegen. Über die gesamte Testreihe hinweg lagen die Meßwerte je nach anliegender Nutzerlast gleichmäßig gestreut im Bereich zwischen diesen Extremwerten, wobei die besseren Werte erwartungsgemäß nachts ermittelt wurden.
Die Benchmarks 110.applu und 146.wave5 zeigen, daß sie besonders speicherhungrig sind, bei ihnen liegen die SYS-Zeiten im Bereich 2000 bzw. 200 Sekunden, während diese auf anderen Maschinen mit viel Speicher im Bereich unter 10 Sekunden liegen.
Aus den folgenden Werten, die mit dem Unix-Kommando time für den Benchmark 110.applu gemessen wurden, kann man gut den Einfluß verschiedener Lastverhältnisse auf die Gesamtlaufzeit und die CPU-Zeiten bei der Programmabarbeitung ablesen:

```
(a)/a/zuse-f/home/urz/fs22/c/cmun/Projekt/HERK/benchspec/110.applu/time.out
    41114.9 real       2650.8 user       1792.4 sys
HERK: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)

(b)/a/zuse-f/home/urz/fs22/c/cmun/Projekt/HERK/benchspec/110.applu/time.out
    46844.7 real       2682.4 user       1762.6 sys
HERK: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)

(c)/a/zuse-f/home/urz/fs22/c/cmun/Projekt/HERK/benchspec/110.applu/time.out
    42398.6 real       2633.1 user       2115.6 sys
HERK: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)
```

Messung (a) ist jene, die in 3.1.2. dokumentiert ist. Aus dieser und den beiden anderen Messungen kann man nun einige Aussagen über die Last auf der Maschine während der Messungen ableiten.
So liegt bei (b) die SYS-Zeit unter jener von (a), die Gesamtlaufzeit jedoch mehr als eine Stunde darüber, es liegt der Schluß nahe, daß sich einige Prozesse mehr die Rechenzeit der CPU teilen mußten, die jedoch insgesamt nicht mehr Speicher benutzen, als die Prozesse bei Messung (a), da die primär durch das Paging beeinflußte SYS-Zeit nicht angestiegen ist.
Der gegenteilige Effekt ist bei Messung (c) zu verzeichnen. Hier ist die Gesamtzeit wesentlich geringer als bei bei (b), die SYS-Zeit jedoch wesentlich höher. Die läßt auf weniger Prozesse mit jedoch höherem Speicherbedarf schließen. Beachtet man, daß eine Pagingoperation einen Zeitaufwand im Bereich einiger Millisekunden beansprucht, so ist der Gesamtaufwand für Speicherauslagerungen hier schon erheblich gestiegen, wodurch auch das insgesamt große Speicherdefizit auf dem Herkules sehr gut verdeutlicht wird.
Der Paging-Effekt läßt sich auch an den Zeiten einiger anderer Benchmarks vermuten. Da er dort jedoch nicht in dieser Größenordnung verzeichnet wird, läßt er sich nicht so eindeutig von eventuellen anderen Einflußfaktoren trennen, so daß man für den Nachweis isolierte Messungen machen müßte, die man durch künstlich generierte Lasten für Prozessor und/oder Speicher unterlegt.

Die Optimierung mußte auf dieser Maschine auf den Schalter -O4 beschränkt werden, der außerdem nicht beim Linken verwendet werden darf. Dadurch ist der Zeitgewinn gegenüber der Standardoptimierung -O (default --> O3) auch gering.
Der -fast Schalter der SPEC-Referenzmessung auf SPARCstation 10/40 generiert hier einen Objektcode, der anschließend vom Linker wegen der Benutzung eines undefinierten Symbols (cg92_used) abgelehnt wird. Die

Versuche mit einzelnen Flags (-cg92, -cg89) dieses Sammelschalters führten nur zu schlechteren Meßergebnissen, so daß auf weitere Optimierungsversuche (auch wegen mangelnder Vergleichbarkeit) verzichtet wurde.
Wegen der starken Beeinflußbarkeit der Messungen durch unterschiedliche Lastbedingungen ist der Zeitgewinn durch optimierte Übersetzung nur an an USER- und SYS-Zeiten ablesbar.

Als Gesamtfazit bleibt nur zu sagen, daß der Rechner herkules.hrz wohl dringend einer Erweiterung des Hauptspeichers bedarf, um seinen Aufgaben als Computeserver bei den immer speicheraufwendiger werdenden Applikationen auch in Zukunft gerecht zu werden.

3.2. SPARCstation 20 Modell 612 (samson.hrz.tu-chemnitz.de)

3.2.1. Gerätekonfiguration

Hardware:

Modell:	SPARCstation 20 Modell 612
Hersteller:	Sun Microsystems, Inc.
CPU:	Superskalar SPARC 60 MHz
Anzahl CPU's:	2
FPU:	Integriert
Primärcache:	20 KB(I) + 16 KB(D) on chip pro CPU
Sekundärcache:	1 MB(D+I) Supercache off chip pro CPU
Hauptspeicher:	128 MB
Externe Speicher:	
Besonderheiten:	Multiprozessormaschine

Software:

Betriebssystem:	SunOS 5.3
	(Solaris 2.3)
Compiler:	Sun FORTRAN 3.0.1
	Sun C 3.0.1
Filesystem:	NFS
Systemstatus:	Multi-User (Tests auch unter echten Multi-User-Bedingungen gelaufen)

weitere Daten:

IP-Name:	samson.hrz.tu-chemnitz.de
IP-Nummer:	134.109.132.6 Ethernet
	134.109.2.6 FDDI
Standort:	Straße der Nationen

Diese Daten stammen direkt aus dem Universitätsrechenzentrum beziehungsweise aus den Veröffentlichungen des URZ im WWW .
(Adresse: http://www.tu-chemnitz.de/home/czi/admin/hosts/samson.html)

3.2.2. Resultate der Messungen

a) SPECfp95 :

Die folgenden Ergebnisse wurden entsprechend der SPEC-Regeln (SPEC-Run-Rules [7]) ermittelt. Es werden die tatsächlichen Laufzeiten der einzelnen Benchmarks betrachtet. Die Auswertung orientiert sich an den SPEC-Regeln zur Veröffentlichung offizieller Ergebnisse (SPEC-Reporting-Rules [8]).

BENCHMARK	SPEC-REFERENZ-ZEIT SEKUNDEN	LAUFZEIT SEKUNDEN	SPECratio	
101.tomcatv	3700	1789,9	2,1	
102.swim	8600	1428,2	6,0	
103.su2cor	1400	1474,6	0,9	
104.hydro2d	2400	5742,0	0,4	
107.mgrid	2500	2538,0	1,0	
110.applu	2200	2424,1	0,9	
125.turb3d	4100	364,7	11,2	
141.apsi	2100	1450,4	1,4	*
145.fpppp	9600	6440,5	1,5	
146.wave5	3000	1750,0	1,7	

Geometrisches Mittel (SPECfp95): 1,66* (Keine Compiler-Optimierung)

Zusätzlich werden nachfolgende Zeiten betrachtet, die bei jedem Testlauf quasi als Nebenprodukt ermittelt werden. In offiziellen SPEC-Veröffentlichungen sind diese Daten nicht enthalten, sie stellen also nur eine Interpretationshilfe für die speziellen Anwendungsbedingungen in diesem Fall dar. Es handelt sich dabei um die USER- und SYS-Zeiten, also die Zeit, in der ein Benchmark tatsächlich einer CPU zugeteilt war.

BENCHMARK	SPEC-REFERENZ-ZEIT SEKUNDEN	CPU-ZEIT SEKUNDEN (Summe von USER-und SYS-Zeit)	USratio	
101.tomcatv	3700	1548,5	2,4	
102.swim	8600	1246,1	6,9	
103.su2cor	1400	1298,3	1,1	
104.hydro2d	2400	4986,0	0,5	
107.mgrid	2500	2420,0	1,0	
110.applu	2200	2113,5	1,0	
125.turb3d	4100	337,5	12,1	
141.apsi	2100	768,7	2,7	*
145.fpppp	9600	5642,5	1,7	
146.wave5	3000	1532,7	2,0	

Geometrisches Mittel US_fp95: 2,00* (Keine Compiler-Optimierung)

SPECfp95/US_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Der Benchmark 141.apsi bricht vorzeitig ab. In der Fließkommaarithmetik tritt ein Fehler (Ungenauigkeit oder Unterlauf) auf, der nicht behoben werden kann. Bezüglich dieses Fehlers wurde bei Reinhold Weicker [1] als SPEC-Ansprechperson in Europa angefragt, woraufhin mitgeteilt wurde, daß das Problem bekannt sei und in der offiziellen Release behoben sein wird. Der vorzeitige Abbruch verfälscht den Wert für SPECratio leicht nach oben.

Flags: OPT = -O
EXTRA_FFLAGS = -O

Bei den Optimierten Meßreihen ergaben sich folgende Ergebnisse:

BENCHMARK	SPEC-REFERENZ-ZEIT SEKUNDEN	LAUFZEIT SEKUNDEN	SPECratio base	
101.tomcatv	3700	1789,9	2,1	
102.swim	8600	1428,2	6,0	
103.su2cor	1400	1474,6	0,9	
104.hydro2d	2400	5742,0	0,4	
107.mgrid	2500	2538,0	1,0	
110.applu	2200	2424,1	0,9	
125.turb3d	4100	364,7	11,2	
141.apsi	2100	1450,4	1,4	*
145.fpppp	9600	6440,5	1,5	
146.wave5	3000	1750,0	1,7	

Geometrisches Mittel SPECfp95_base : 1,66* (Baseline_Optimierung)

Die Werte sind mit denen der nichtoptimierten Messung identisch, der Grund dafür wird im Punkt 3.2.3. kurz erläutert. Trotzdem bewirken die Optimierungsparameter eine Beschleunigung der Abarbeitung. Bei der reinen Laufzeitmessung ist diese aufgrund der permanent hohen Last nicht sichtbar. Die tatsächliche Verbesserung wird nur bei Betrachtung der reinen Prozessorzeit (USER+SYS-Zeit) gegenüber der Standardmessung deutlich.

BENCHMARK	SPEC-REFERENZ-ZEIT SEKUNDEN	CPU-ZEIT SEKUNDEN (Summe von USER-und SYS-Zeit)	USratio base	
101.tomcatv	3700	1390,2	2,7	
102.swim	8600	1243,2	6,9	
103.su2cor	1400	1201,1	1,2	
104.hydro2d	2400	4641,1	0,5	
107.mgrid	2500	1632,2	1,5	
110.applu	2200	1712,7	1,3	
125.turb3d	4100	261,5	15,7	
141.apsi	2100	630,5	3,3	*
145.fpppp	9600	3071,0	3,1	
146.wave5	3000	1192,5	2,5	

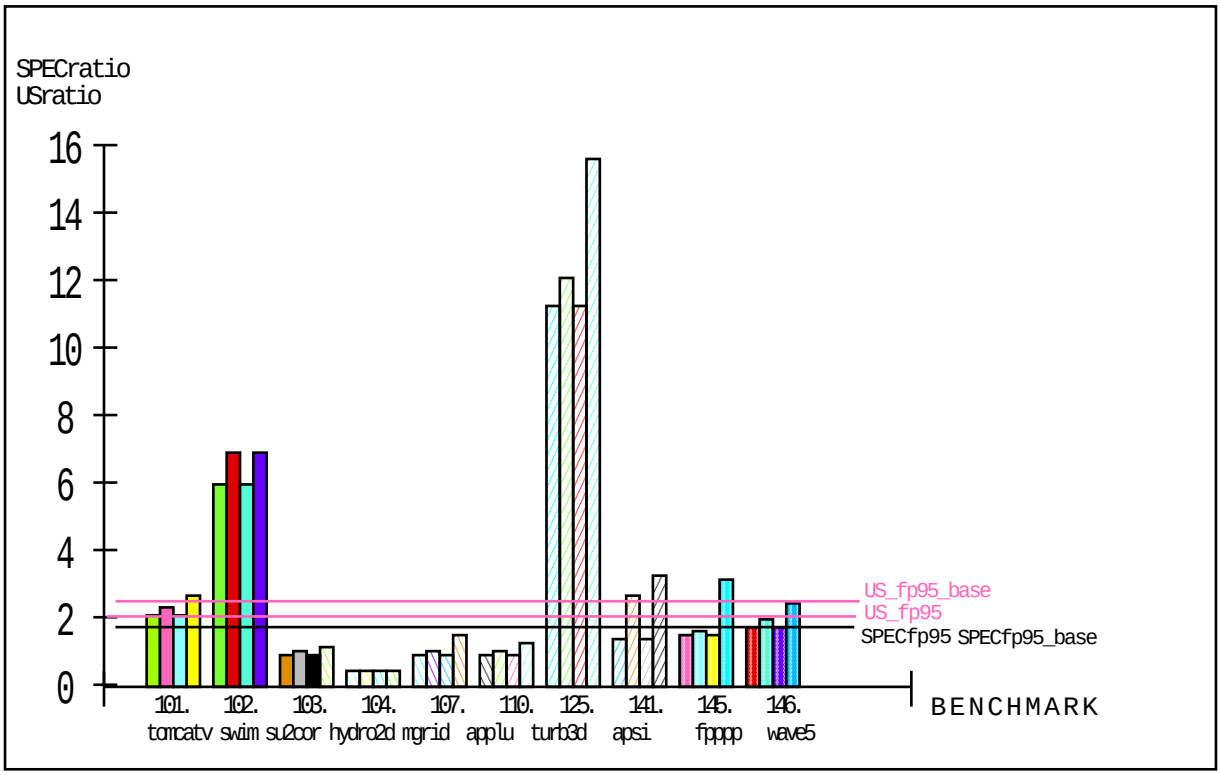
Geometrisches Mittel US_fp95_base : 2,48* (Baseline-Optimierung)

SPECfp95/US_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Der schon erläuterte Fehler bei 141.apsi ließ sich auch mit verschiedenen in Frage kommenden Flags nicht beseitigen .

Flags: OPT = -O4
EXTRA_FFLAGS = -fast -O4

Im folgenden Diagramm werden für jeden Benchmark die 4 Meßwerte nochmals gegenübergestellt.



Die vier Säulen zeigen jeweils die Werte SPECratio,
USratio,
SPECratio(baseline_optimiert),
USratio(baseline_optimiert)

in eben dieser Reihenfolge für jeden der 10 Benchmarks.

Die vier Querlinien zeigen die Gesamtmittelwerte SPECfp95, SPECfp95_base (hier identisch) und die zusätzlichen "Grenzwerte" US_fp95 und US_fp95_base.

b) SPECrate_fp95:

Der Rechner Samson ist der einzige im Testfeld, der über mehrere Prozessoren verfügt und für den daher eine spezielle Durchsatz-Messung erforderlich ist. Für jede CPU wird üblicherweise eine Kopie des jeweiligen Benchmarks gestartet, in diesem Fall laufen also 2 Kopien gleichzeitig ab. Bei der besten Messung ergaben sich folgende Ergebnisse:

BENCHMARK	REF-ZEIT SEKUNDEN	ANZAHL KOPIEN	ZEIT SEKUNDEN	SPECrate
101.tomcatv	3700	2	1789,90	37,21
102.swim	8600	2	1428,20	108,39
103.su2cor	1400	2	1474,60	17,09
104.hydro2d	2400	2	5742,00	7,52
107.mgrid	2500	2	2538,00	17,73
110.applu	2200	2	2424,10	16,34
125.turb3d	4100	2	364,70	202,36
141.apsi	2100	2	1450,40*	26,06*
145.fpppp	9600	2	6440,50	26,83
146.wave5	3000	2	1750,00	30,86

Geometrisches Mittel (SPECrate_fp95): 30,32*

SPECrate_fp95/USrate_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Für den Fehler bei 141.apsi treffen die Bemerkungen aus 3.2.2. a) zu.

Flags: OPT = -O
EXTRA_FFLAGS = -O

Analog zu den Laufzeitmessungen wurde eine optimierte Durchsatzmessung durchgeführt.

Flags: OPT = -O4
EXTRA_FFLAGS = -fast -O4

Wegen der höheren Last wurden aber längere REAL-Zeiten gemessen, so daß auf die Dokumentation hier verzichtet wird. An den USER/SYS-Werten kann jedoch eine Beschleunigung in etwa in der gleichen Größenordnung wie bei den einfachen Laufzeitmessungen festgestellt werden. Auf die Darstellung der Einzelzeiten kann verzichtet werden.
Weitere Erklärungen finden sich in 3.2.3.

3.2.3 Erläuterung und Interpretation der SAMSON-Messungen

Der Rechner samson.hrz.tu-chemnitz.de verfügt als einziger im Testfeld über mehrere Prozessoren, genauer gesagt über deren 2. Daher ist es nicht verwunderlich, daß er bei den Durchsatz-Messungen ("SPECrate") die besten (offiziellen) Werte aller Teilnehmer erreicht.
Nur für diesen Rechner ist überhaupt eine spezielle Messung mit jeweils 2 gleichen Programmkopien erforderlich gewesen. Begünstigt durch die gute Bestückung mit Hauptspeicher können 2 Programmkopien tatsächlich in der gleichen Zeit ausgeführt werden, wie ein einzelnes Programm, sofern die CPU´s nicht durch andere rechenzeitintensive Anwendungen zusätzlich (evtl. sogar ungleichmässig) belastet werden.
Die Meßergebnisse für SPECfp95 stammen aus einer solchen Durchsatz-Messung mit 2 Programmkopien. Sie wurden kurz nach einem Neustart ermittelt, als auf dem Server einmal (fast) keine weiteren Aktivitäten zu verzeichnen waren. Die Bedingungen sind somit schon eher mit den sonst für die Messungen bei den Herstellen üblichen Bedingungen zu vergleichen. Leider konnte zum damaligen Zeitpunkt nicht gleich noch eine baseline-optimierte Messung durchgeführt werden. Es ist durchaus als legitim zu bezeichnen, die gemessenen Laufzeiten bei 2 Kopien auch als gültige Werte für die einfachen Laufzeitmessungen SPECfp zu betrachten, da die Ausführungszeit für eine Kopie nur noch kürzer sein könnte.
Die optimierten Ergebnisse sind bei dieser Arbeit stets die besten überhaupt gemessenen Werte. Die Gleichheit von optimierten und nichtoptimierten Ergebnissen hier liegt darin begründet, daß bei den Versuchen mit den Optimierungsflags die guten Ergebnisse der oben erwähnten Messung wegen höherer Rechenlast nicht wieder erreicht wurden.
Unter "normalen" Bedingungen (also mit hoher Last) lagen die Gesamtergebnisse für SPECfp95 im Schnitt bei 0.7 und für die optimierte Variante SPECfp95_base in etwa bei 1. Diese eher durchschnittlich bis schlecht anmutenden Ergebnisse sind durchaus akzeptabel, wenn man betrachtet, wieviele Anwendungen oft gleichzeitig aktiv sind. Neben den typischen X-Anwendungen, WWW-Browsern und diversen Netzprogrammen sind häufig rechenintensive Simulationen vertreten (wie man zumindest aus den Namen schließen kann), deren Ausgangspunkte hauptsächlich in den Fachbereichen ET und Mathematik liegen. Speziell diese Simulationen werden auch die FPU´s des Rechners benutzen. An den SYS-Zeiten, die nie über 10 Sekunden lagen, läßt sich ablesen, daß die Speicherausstattung ausreichend ist, da "zeitraubende" Paging-Aktivitäten nur selten auftreten.
Bei der Menge der Anwendungen ist es sehr unwahrscheinlich, daß die relativ großen Zeitunterschiede zwischen Gesamtzeit und USER+SYS-Summe durch Wartezeit bei I/O-Operationen hervorgerufen werden (etwa durch langsame Netzverbindungen oder externe Speichermedien). Es ist eher anzunehmen, daß die von Hause aus guten Durchsatzleistungen dieser Maschine durch die vielen Nutzer aus diversen Fachbereichen und die CSN-Nutzer in den Wohnheimen (speziell auch in den eigentlich verkehrsärmeren Abendstunden) nur zu gern ausgelastet werden.
Die bei SPECrate-Messungen ebenfalls ermittelbaren reinen CPU-Zuteilungszeiten sind weniger aussagekräftig, da sie nur als Summe für alle Programmkopien zur Verfügung stehen. Man könnte zwar zumindest durch Division durch die Prozessorenzahl einen Mittelwert bilden, da die USER/SYS-Zeiten jedoch für offizielle SPEC-Ergebnisse sowieso nicht verwendet werden dürfen, wurde auf diese weitere "Verkünstlichung" der Ergebnisse verzichtet. Statt dessen wurden für die "Hilfs"ergebnisse US_fp95 und US_fp95_base die besten USER/SYS-Zeiten verwendet, die tatsächlich bei Messungen mit einem Benchmark-Exemplar ermittelt wurden.
An diesen Ergebnissen läßt sich dann auch ablesen, daß durch die Flags -O4 und -fast tatsächlich eine Verbesserung gegenüber -O erreicht wird.
Durch -O wird die Standardoptimierung beim Compilerlauf aktiviert. Diese entspricht beim samson der Stufe O3 und realisiert lokale und globale Optimierung (Sprünge, Schleifen..) und die Auflösung externer Variablenvereinbarungen und Referenzen.
Bei den optimierten Messungen wird durch -fast eine Reihe zusätzlicher Optimierungen ausgeführt, nämlich

-cg89 bzw. -cg92	(schnellsten Code für entsprechende aktuelle Hardware)
-O3	
-fnonstd	(spezielle Initialisierung der FP-Arithmetik für Ausnahmebeh.)
-fsimple	(einfaches FP-Arithmetik-Modell)
-dalign	(spezielle Ausrichtung im Speicher)
-xlibmopt	

Der Schalter -O4 ist eigentlich unnütz, da er durch -fast automatisch wieder mit -O3 überlagert wird. Daran kann man sehen, daß bei den Messungen von SUN, aus denen die Schalter übernommen wurden, offensichtlich eine andere Compilerversion verwendet wurde.
Mit diesen und weiteren Schaltern für spezielle arithmetische Probleme (-fieee, -fnonstd) ist es während der Tests nicht gelungen, das Problem der vorzeitigen Terminierung von 141.apsi aufgrund eines Fehlers in der

Fließkommaarithmetik zu beseitigen. Im Verzeichnis existiert auch noch kein SUN-spezifisches Sourcefile. (Das gleiche Problem trat auch schon auf dem anderen SUN-Rechner Herkules auf).
Der aus den USER/SYS-Zeiten für optimierte/nichtoptimierte Messungen ablesbare Beschleunigungsfaktor ist in etwa gleich, egal ob man die Zeiten aus der reinen Laufzeitmessung oder die Gesamtsummen für die Durchsatzmessungen betrachtet.
Bei den Durchsatzmessungen lassen sich auch Informationen über den Einfluß von Befehls-Caches gewinnen. So ist zu verzeichnen, daß die Summe der USER/SYS-Zeiten für zwei Kopien stets etwas unter dem verdoppelten Wert der Einzelmessungen liegt, was darauf zurückzuführen ist, daß der Code praktisch nur einmal geladen werden muß und für den Prozessor ,der gerade "nachhängt", mit hoher Wahrscheinlichkeit im Instruction-Cache vorhanden ist. Dabei wird vorausgesetzt, daß die Kopien auf allen CPU's in etwa gleich schnell ausgeführt werden. Dies ist wiederum von der Lastverteilung abhängig. Es ist jedoch nicht möglich, hieraus konkrete Aussagen über die konkrete Cache-Konfiguration eines Rechners abzuleiten. Dafür ist der Zeitgewinn im Verhältnis zur Gesamtausführungszeit zu gering, durch den größeren Einfluß anderer Ressourcen können außerdem wesentlich stärkere Ergebnisveränderungen leicht zu einer Fehlinterpretation des Cache-Einflusses führen. Ein Beispiel zur Illustration dieses Cache-Effektes ist im Kapitel 4 angegeben.

Vergleich mit offiziellen Veröffentlichungen von Meßergebnissen:

Eine Veröffentlichung offizieller SPECfp95- beziehungsweise SPECrate_fp95-Ergebnisse für eine SPARCstation 20 Modell 612 ist noch nicht verfügbar. Die Meßergebnisse für eine vergleichbare Maschine mit der Testsuite CFP92 finden sich auf dem WWW-Server von SPEC [2]

SPECfp92: http://performance.netlib.org/performance/html/spec.sun20612.cfp92.9_94.notes.html#
{Sun MicrosystemsInc. SPARCstation/server 20 Model 612 60MHz SuperSPARC}
SPECbase_fp92 = 111.0

SPECrate_fp92: http://performance.netlib.org/performance/html/spec.sun20612.crfp92.6_94.notes.html#
{Sun Microsystems Inc. SPARCstation/server 20 Model 612 60MHz SuperSPARC}

Auf diesem Server finden sich auch SPECfp92-Resultate für die neue Referenzmaschine bei SPECfp95 (SPARCstation 10/40), so daß man die Ergebnisse zumindest in Relation setzen kann:

SPECfp92: (http://performance.netlib.org/performance/html/spec.sunf41.cfp92.6_93.notes.html#
{Sun Microsystems Inc. SPARCstation 10 Model 40 40MHz TMS390Z50})
Geometric Mean (SPECfp92): 60.2

Setzt man die Ergebnisse in Relation, so erhält man einen Faktor von 1,84. Obwohl die Ergebnisse nicht direkt umgerechnet werden können sind, zeigt dies, daß die in den Versuchen mit der CFP95-Suite ermittelten Werte durchaus im realistischen Bereich liegen. So liegt dieser Faktor 1,84 genau zwischen dem ermittelten SPECfp95_base (1,66) und den "theoretischen Grenzwerten" US_fp95 (2.0) bzw. US_fp95_base (2,48).
Die Durchsatzwerte für die 92er und die 95er Suite sind nicht vergleichbar, da sich bei deren Berechnung Normalisierungsfaktoren und Bezugszeitraum unterscheiden.

3.3. RW420-XS (IRIS Indigo) -(R4000 100 Mhz) (silly.hrz.tu-chemnitz.de)

3.3.1. Gerätekonfiguration

Hardware:

Modell:	RW420-XS (IRIS Indigo)
Hersteller:	Silicon Graphics, Inc.
CPU:	R4000 100 MHz
Anzahl CPU's:	1
FPU:	Integriert
Primärcache:	8 KB on chip
Sekundärcache:	1 MB(I+D) off chip
Hauptspeicher:	48 MB
Externe Speicher:	HD 2GB (intern) HD 540 MB (intern)

Software:

Betriebssystem:	SGI IRIX 5.2
Compiler:	SGI FORTRAN Compiler SGI C Compiler
Filesystem:	NFS
Systemstatus:	Multi-User (Tests auch unter echten Multi-User-Bedingungen gelaufen)

weitere Daten:

IP-Name:	silly.hrz.tu-chemnitz.de
IP-Nummer:	134.109.200.34 Ethernet
Standort:	Straße der Nationen, Böttcherbau, Raum 071

Diese Daten stammen direkt aus dem Universitätsrechenzentrum beziehungsweise aus den Veröffentlichungen des URZ im WWW .
(Adresse: http://www.tu-chemnitz.de/home/ghe/MASCHINEN/indigo)

3.3.2. Resultate der Messungen

a) SPECfp95 :

Die folgenden Ergebnisse wurden entsprechend der SPEC-Regeln (SPEC-Run-Rules [7]) ermittelt, das heißt, es werden die tatsächlichen Laufzeiten der einzelnen Benchmarks betrachtet. Die Auswertung orientiert sich an den SPEC-Regeln zur Veröffentlichung offizieller Ergebnisse (SPEC-Reporting-Rules [8]).

BENCHMARK	SPEC-REFERENZ-ZEIT SEKUNDEN	LAUFZEIT SEKUNDEN	SPECratio	
101.tomcatv	3700	2689,4	1,4	
102.swim	8600	1412,3	6,1	*
103.su2cor	1400	1325,5	1,1	
104.hydro2d	2400	4620,4	0,5	
107.mgrid	2500	2088,4	1,2	
110.applu	2200	1931,9	1,1	
125.turb3d	4100	260,2	15,8	
141.apsi	2100	1587,5	1,3	
145.fpppp	9600	9138,4	1,1	*
146.wave5	3000	2820,4	1,1	

Geometrisches Mittel (SPECfp95): 1,65* (Keine Optimierung)

Zusätzlich werden nachfolgende Zeiten betrachtet, die bei jedem Testlauf quasi als Nebenprodukt ermittelt werden. In offiziellen SPEC-Veröffentlichungen sind diese Daten nicht enthalten, sie stellen nur eine Interpretationshilfe für die speziellen Anwendungsbedingungen in diesem Fall dar. Es handelt sich dabei um die USER- und SYS-Zeiten, also die Zeit, in der ein Benchmark tatsächlich einer CPU zugeteilt war.

BENCHMARK	SPEC-REFERENZ-ZEIT	GEMESSENE ZEITEN	STD	MIN	SEK	SUMME USER+SYS SEKUNDEN	USratio	
101.tomcatv	3700	USER		43	33,9	2619,9	1,4	
		SYS			6,0			
102.swim	8600	USER		23	14,2	1397,2	6,2	*
		SYS			3,0			
103.su2cor	1400	USER		21	28,5	1293,8	1,1	
		SYS			5,3			
104.hydro2d	2400	USER	1	15	34,7	4548,7	0,5	
		SYS			14,0			
107.mgrid	2500	USER		34	5,6	2051,6	1,2	
		SYS			6,0			
110.applu	2200	USER		31	34,6	1903,3	1,2	
		SYS			8,7			
125.turb3d	4100	USER		4	15,6	257,3	15,9	
		SYS			1,7			
141.apsi	2100	USER		18	23,1	1522,6	1,4	
		SYS		6	59,5			
145.fpppp	9600	USER	2	25	12,3	8728,4	1,1	*
		SYS			16,1			
146.wave5	3000	USER		25	48,6	1674,6	1,8	
		SYS		2	6,0			

Geometrisches Mittel US_fp95: 1,77* (KeineOptimierung)

SPECfp95/US_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Bei beiden mit * gekennzeichneten Benchmarks laufen bis zum Ende durch. Beim anschließenden Ergebnisvergleich mit den SPEC-Referenzergebnissen findet das Vergleicherprogramm spiff jedoch Unterschiede zwischen den Referenzergebnissen $SPECHOME/benchspec/$BENCHMARK/result.ref/$BENCHMARK.in.out und den berechneten Ergebnissen $SPECHOME/benchspec/$BENCHMARK/resultre/clone_0/$BENCHMARK.in.out. Ein zertifiziertes, alternatives Ergebnisfile (....in.out.alt) für die SGI-Plattform findet sich für keinen der

Benchmarks, ebensowenig ein alternativer Benchmarkquelltext für SGI. Die Lösung des Problems liegt schon in den Optimierungsschaltern, wie in den anschließenden Experimenten mit den Compilerflags festgestellt werden konnte.

Flags: OPT = -O

Bei den Compileroptimierten Meßreihen ergaben sich folgende Ergebnisse:

BENCHMARK	SPEC-REFERENZ-TIME SEKUNDEN	LAUFZEIT SEKUNDEN	SPECratio base	
101.tomcatv	3700	2485,0	1,5	
102.swim	8600	1333,5	6,4	
103.su2cor	1400	1116,0	1,3	
104.hydro2d	2400	4151,5	0,6	
107.mgrid	2500	2083,9	1,2	*
110.applu	2200	1740,5	1,3	
125.turb3d	4100	202,1	20,3	
141.apsi	2100	1459,0	1,4	
145.fpppp	9600	6892,8	1,4	
146.wave5	3000	2475,4	1,2	

Geometrisches Mittel (SPECfp95_base): 1,88 (Baseline-optimiert)

Die Werte bei Testläufen mit Optimierungsschaltern sind etwas besser als die der nichtoptimierten Messungen. Da auf dem Rechner Silly meist keine allzu große Last vorhanden ist (Ausnahmen bestätigen die Regel), kann man die Laufzeiten einigermaßen vergleichen, die Optimierung kommt sowohl in den reinen Laufzeiten als auch in den Zuteilungszeiten (USER+SYS) zu Ausdruck. Viel wichtiger ist, daß die Differenzen in den Resultaten durch die verwendeten Schalter beseitigt werden.

BENCHMARK	SPEC-REFERENZ-ZEIT	GEMESSENE ZEITEN STD	MIN	SEK	SUMME USER+SYS SEKUNDEN	USratio base	
101.tomcatv	3700	USER	40	42,4	2448,0	1,5	
		SYS		5,6			
102.swim	8600	USER	21	53,8	1316,6	6,5	
		SYS		2,8			
103.su2cor	1400	USER	18	4,3	1089,9	1,3	
		SYS		5,6			
104.hydro2d	2400	USER 1	7	51,8	4085,1	0,6	
		SYS		13,3			
107.mgrid	2500	USER	34	5,6	2051,6	1,2	*
		SYS		6,0			
110.applu	2200	USER	28	24,5	1711,5	1,3	
		SYS		7,0			
125.turb3d	4100	USER	3	17,8	199,3	20,6	
		SYS		1,5			
141.apsi	2100	USER	16	46,6	1424,3	1,5	
		SYS	6	57,7			
145.fpppp	9600	USER 1	52	50,0	6782,3	1,4	
		SYS		12,3			
146.wave5	3000	USER	21	21,1	1387,5	2,2	
		SYS	1	46,4			

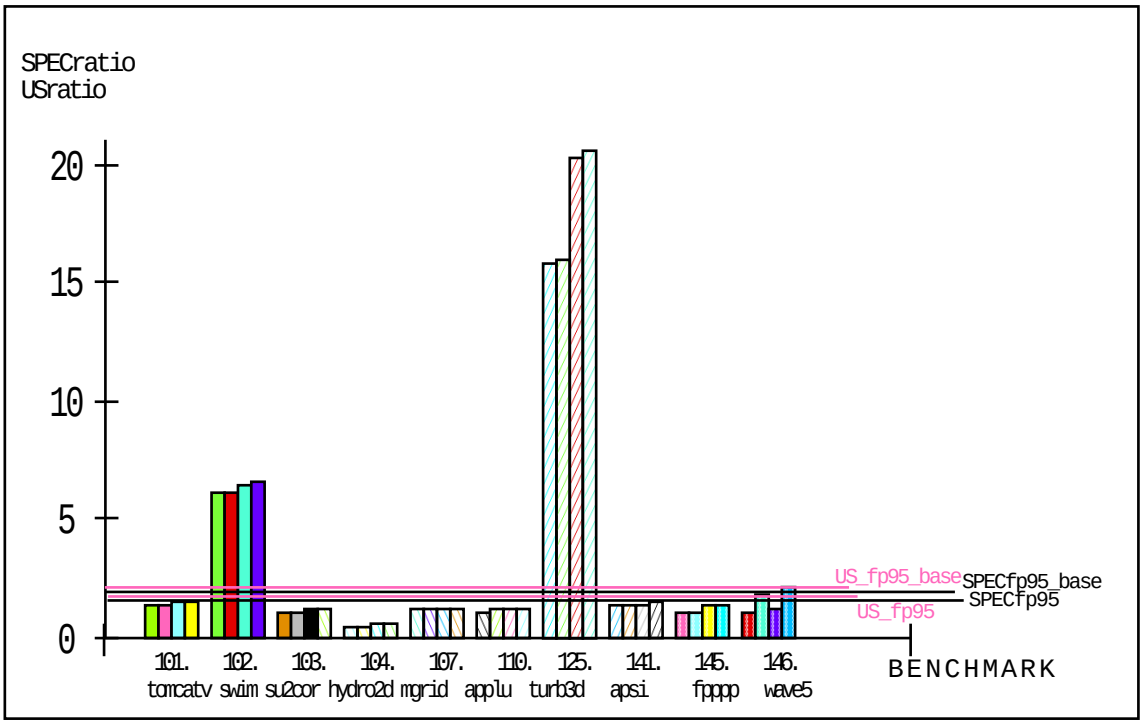
Geometrisches Mittel US_fp95_base: 2,02 (Baseline-optimiert)

SPECfp95/US_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

Flags: OPT = -O -mips2 -non_shared (-jmpopt -Nn6000 -Olimit 2000)

!! Ausnahme *: Beim Benchmark 107.mgrid führt diese Schalterkombination zum schnellen Abbruch des Benchmarks, deshalb wurde bei diesem wieder nur -O verwendet.

Im folgenden Diagramm werden für jeden Benchmark die 4 Meßwerte nochmals gegenübergestellt.



Die vier Säulen zeigen jeweils die Werte SPECratio,
USratio,
SPECratio(baseline_optimiert),
USratio(baseline_optimiert)

in eben dieser Reihenfolge für jeden der 10 Benchmarks.

Die vier Querlinien zeigen die Gesamtmittelwerte SPECfp95, SPECfp95_base und die zusätzlichen "Grenzwerte" US_fp95 und US_fp95_base.

b) SPECrate_fp95:

Da der Rechner Silly nur eine CPU besitzt, ist keine spezielle Durchsatzmessung erforderlich, die Werte für SPECrate_fp95 ergeben sich durch Berechnung aus den Werten der Laufzeitmessungen in Teil a) :
(Für den ergänzenden "Grenzwert" USrate ist an die Tabelle eine zusätzliche Spalte angefügt, die Werte sind jeweils aus der Summe aus USER- und SYS-Zeiten der Laufzeitmessungen berechnet, die hier nicht nochmals einzeln gelistet werden.)

BENCHMARK	REF-ZEIT SEKUNDEN	ANZAHL KOPIEN	ZEIT SEKUNDEN	SPECrate	USrate	
101.tomcatv	3700	1	2689,4	12,4	12,7	
102.swim	8600	1	1412,3	54,8	55,4	*
103.su2cor	1400	1	1325,5	9,5	9,7	
104.hydro2d	2400	1	4620,4	4,7	4,7	
107.mgrid	2500	1	2088,4	10,8	11,0	
110.applu	2200	1	1931,9	10,2	10,4	
125.turb3d	4100	1	260,2	141,8	143,4	
141.apsi	2100	1	1587,5	11,9	12,4	
145.fpppp	9600	1	9138,4	9,5	9,9	*
146.wave5	3000	1	2820,4	9,6	16,1	
Geometrisches Mittel	(SPECrate_fp95):			14,8*		
	(USrate_fp95):				15,9*	

SPECrate_fp95/USrate_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Für die Fehler bei 102.swim und 145.fppp treffen die Bemerkungen aus 3.3.2. a) zu.

Flags: OPT = -O

Aus den Werten der optimierten Laufzeitmessungen ergeben sich folgende Durchsatzwerte:

BENCHMARK	REF-ZEIT SEKUNDEN	ANZAHL KOPIEN	ZEIT SEKUNDEN	SPECrate base	USrate base	
101.tomcatv	3700	1	2485,0	13,4	13,6	
102.swim	8600	1	1333,5	58,0	58,8	
103.su2cor	1400	1	1116,0	11,3	11,6	
104.hydro2d	2400	1	4151,5	5,2	5,3	
107.mgrid	2500	1	2083,9	10,8	11,0	*
110.applu	2200	1	1740,5	11,4	11,6	
125.turb3d	4100	1	202,1	182,6	185,1	
141.apsi	2100	1	1459,0	13,0	13,3	
145.fpppp	9600	1	6892,8	12,5	12,7	
146.wave5	3000	1	2475,4	10,9	19,5	
Geometrisches Mittel	(SPECrate_fp95_base):			16,8		
	(USrate_fp95_base):				18,1	

SPECrate_fp95/USrate_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

Flags: OPT = -O -mips2 -non_shared (-jmpopt -Nn6000 -Olimit 2000)
!! Ausnahme * : Beim Benchmark 107.mgrid führt diese Schalterkombination zum schnellen Abbruch des Benchmarks, deshalb wurde bei diesem wieder nur -O verwendet.

3.3.3 Erläuterung und Interpretation der SILLY-Messungen

Die Betrachtung der Ergebnisse auf dieser Maschine muß unter anderen Gesichtspunkten erfolgen als bei den 3 Computeservern Samson, Herkules und Tantalus. Silly ist zwar über das Netz für jeden erreichbar, wird aber kaum zur Auslagerung von Anwendungen benutzt. So war es bei den Versuchen des öfteren der Fall, daß außer den üblichen System- und Netzprozessen keine weiteren Anwendungen aktiv waren. Insofern ist es bedauerlich, daß noch keine vergleichbaren CFP95-Ergebnisse von SPEC veröffentlicht wurden, da auf Silly zumindest SPEC-ähnlichere Meßbedingungen vorherrschten, als auf allen anderen getesteten Maschinen. Ein Wertevergleich hätte dabei zeigen können, wie hoch der Einfluß der Ausführung der Ein- und Ausgabeoperationen über das Zwischenmedium "Netzwerk" tatsächlich ist. Dieser Vergleich muß bis zur Veröffentlichung offizieller Meßergebnisse durch SPEC oder durch Silicon Graphics vertagt werden, falls diese für das nicht ganz neue Produkt überhaupt noch erfolgt.

Der Vergleich mit CFP92-Veröffentlichungen ist ein nur bedingt taugliches Instrument. Auf dem WWW-Server von SPEC [2] befindet sich unter der konkreten Adresse

http://performance.netlib.org/performance/html/spec.iscfp.cfp92.9_93.notes.html#
{Silicon Graphics, Inc Indy R4000SC 100MHz MIPS R4000SC}

eine Veröffentlichung für ein System mit zumindest demselben Prozessortyp (R4000 100MHz). Der dort ermittelte Durchschnittswert für SPECfp95 liegt bei 63,0.
Verglichen mit den mit der CFP92-Suite gemessenen Ergebnissen für die neue SPEC -Referenzmaschine SPARCstation 10/40 , veröffentlicht unter der Adresse

(http://performance.netlib.org/performance/html/spec.sunf41.cfp92.6_93.notes.html#
{Sun Microsystems Inc. SPARCstation 10 Model 40 40MHz TMS390Z50})
Geometric Mean (SPECfp92): 60.2

ergibt sich als Verhältnis der Faktor 1. In den Versuchen wurde für die Maschine silly aber ein Wert von SPECfp95_base =1,88 ermittelt. Ursache für die Abweichung ist hauptsächlich der Benchmark 125.turb3d, der wesentlich schneller Abläuft als auf der Referenzmaschine, was auch schon bei den anderen getesteten Maschinen vermuten ließ, daß dessen Laufzeit gegenüber der hier verwendeten Pre-Release CFP95.29 noch erheblich verlängert wurde. Da für die übrigen Benchmarks die SPECratio nahe um 1 liegen, verfälscht der eine "Ausreißer" mit dem Ratio von 15...20 den als Geometrisches Mittel ermittelten Durchschnittswert SPECfp95 erheblich nach oben. Läßt man den 125.turb3d in der Betrachtung außen vor, so liegen SPECfp95 aus den Versuchen und der 92er Vergleichsfaktor wesentlich dichter zusammen. Eine direkte Umrechnung von 92er und 95er Ergebnissen ist sowieso nicht möglich, der Vergleich kann nur in etwa eine Größenordnung liefern. Außerdem wurde bereits erwähnt, daß die Maschine lediglich über denselben Prozessortyp verfügt. Durchsatzwerte für die 92er und die 95er Suite sind nicht vergleichbar, da sich bei deren Berechnung Normalisierungsfaktoren und Bezugszeitraum unterscheiden.

Trotz des nicht sehr aussagekräftigen Vergleiches mit den 92er Werten kann man sagen, daß die durch Versuche ermittelten Durchschnittswerte SPECfp95 und SPECfp95_base und die einzelnen SPECratios bei Berücksichtigung der Versionsproblematik relativ "SPEC-nah" sein dürften, was sich aus den geringen Unterschieden zwischen ermittelten SPECfp95 und US_fp95 bzw. den einzelnen SPECratio- und USratio-Werten ablesen läßt.
Die vorhandenen 48 MByte Hauptspeicher stellen bei unbelasteten Maschinen für die CFP95-Benchmarks in etwa die "Schmerzgrenze" dar, die SYS-Zeiten liegen meist unter 10 Sekunden, nur bei 146.wave5 werden ca. 2 Minuten verzeichnet. Auf eine allgemeine Anfrage bzgl. der Speicherproblematik bei Reinhold Weicker [1] meinte er sinngemäß, daß 32 MByte Hauptspeicher eindeutig zu wenig seien, "64 MByte Hauptspeicher sollten schon vorhanden sein". Auch die SPEC-Publikationen zum Thema 95er Benchmarks [3] enthalten die Information, daß 64 MByte Hauptspeicher als derzeitiger Standard und daher als Minimum betrachtet werden. Die hohe SYS-Zeit bei 141.apsi ist schlecht erklärbar, da ein Vergleich mit den SPARC-Maschinen nicht möglich ist, weil dieser Benchmark dort nicht fehlerfrei arbeitete.

Die bei den baseline-optimierten Messungen angegebenen Compilerschalter entstammen der bereits oben angegebenen Veröffentlichung von 92er SPEC-Resultaten auf dem SPEC-WWW-Server. Im einzelnen sollen sie bewirken:

-O	bewirkt -O3 und damit vollständige Codeoptimierung
-mips2	generiert speziellen Code durch Verwendung des speziellen MIPS2-Befehlssatzes und spezielle Bibliotheken
-non_shared	

Die in Klammern angegebenen Flags (-jmpopt -Nn6000 -Olimit 2000) bewirken zusätzliche Sprungoptimierung, Festlegung der maximal möglichen Anzahl lokaler Namen und Begrenzung der Größe zu optimierender Codeblöcke. Deren Einfluß scheint praktisch sehr gering zu sein, bei mehreren Messungen konnten damit nur minimale Veränderungen in positiver wie negativer Richtung festgestellt werden.

3.4. HP Apollo Serie 700 Modell 735/125 (tantalus.hrz.tu-chemnitz.de)

3.4.1. Gerätekonfiguration

Hardware:

Modell:	HP Apollo Serie 700 Modell 735/125
Hersteller:	Hewlett-Packard
CPU:	PA-RISC 125 MHz PA7150
Anzahl CPU's:	1
FPU:	Integriert
Primärcache:	256 KB(I) + 256 KB(D) off chip
Sekundärcache:	-
Hauptspeicher:	144 MB
Externe Speicher:	

Software:

Betriebssystem:	HP-UX 9.01
Compiler:	HP-UX 9.01 FORTRAN 77 Compiler
	HP-UX 9.01 C Compiler
Filesystem:	NFS (HP-UX)
Systemstatus:	Multi-User (Tests auch unter echten Multi-User-Bedingungen gelaufen)

weitere Daten:

IP-Name:	tantalus.hrz.tu-chemnitz.de
IP-Nummer:	134.109.132.3 Ethernet
	134.109.164.3 FDDI
Standort:	Straße der Nationen

Diese Daten stammen direkt aus dem Universitätsrechenzentrum beziehungsweise aus den Veröffentlichungen des URZ im WWW .
(Adresse: http://www.tu-chemnitz.de/home/czi/admin/hosts/tantalus.html)

3.4.2. Resultate der Messungen

a) SPECfp95 :

Die folgenden Ergebnisse wurden entsprechend der SPEC-Regeln (SPEC-Run-Rules [7]) ermittelt. Es werden die tatsächlichen Laufzeiten der einzelnen Benchmarks betrachtet. Die Auswertung orientiert sich an den SPEC-Regeln zur Veröffentlichung offizieller Ergebnisse (SPEC-Reporting-Rules [8]).

BENCHMARK	SPEC-REFERENZ-ZEIT SEKUNDEN	LAUFZEIT SEKUNDEN	SPECratio (base)
101.tomcatv	3700	2221,9	1,7
102.swim	8600	1378,4	6,2
103.su2cor	1400	1968,5	0,7
104.hydro2d	2400	6829,4	0,4
107.mgrid	2500	4279,4	0,6
110.applu	2200	2287,4	1,0
125.turb3d	4100	308,9	13,3
141.apsi	2100	2024,7	1,0
145.fpppp	9600	2524,2	3,8
146.wave5	3000	1300,8	2,3

Geometrisches Mittel (SPECfp95): 1,70

Zusätzlich werden nachfolgende Zeiten betrachtet, die bei jedem Testlauf quasi als Nebenprodukt ermittelt werden. In offiziellen SPEC-Veröffentlichungen sind diese Daten nicht enthalten, sie stellen nur eine Interpretationshilfe für die speziellen Anwendungsbedingungen in diesem Fall dar. Es handelt sich dabei um die USER- und SYS-Zeiten, also die Zeit, in der ein Benchmark tatsächlich einer CPU zugeteilt war.

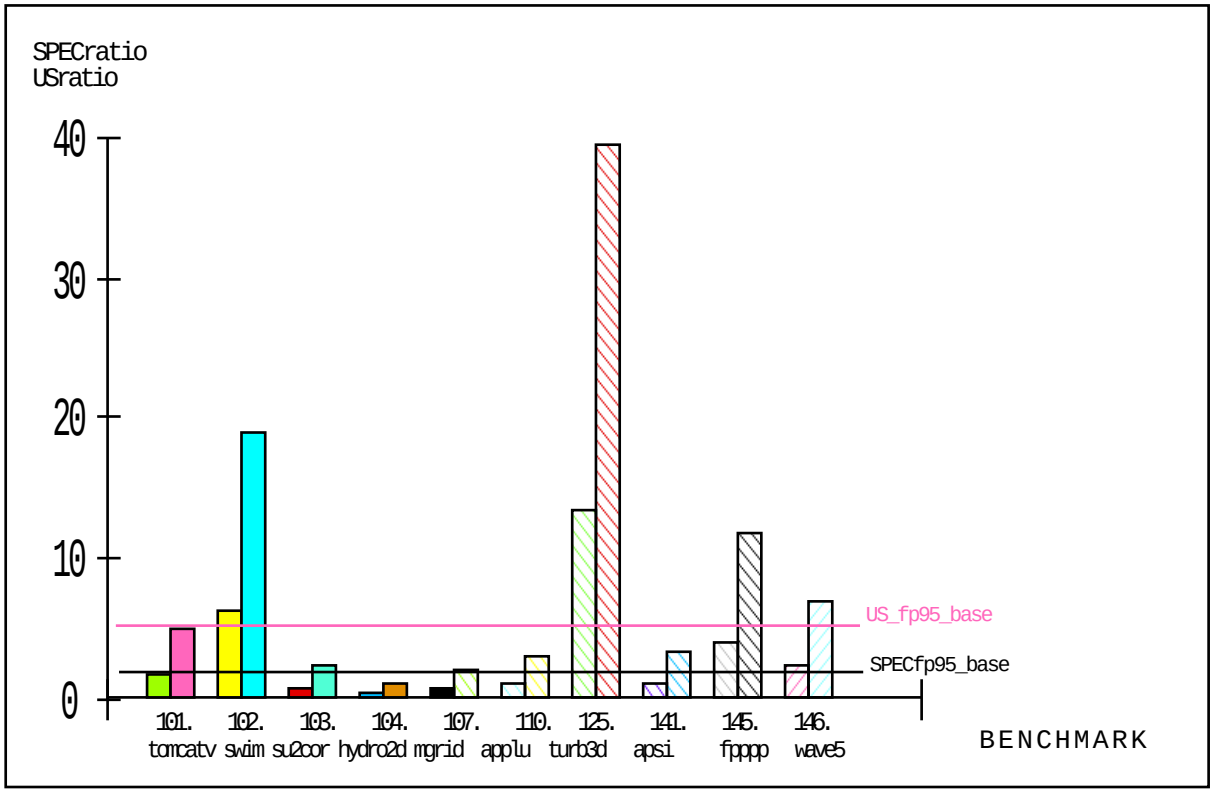
BENCHMARK	SPEC-REFERENZ-ZEIT	GEMESSENE ZEITEN STD	MIN	SEK	SUMME USER+SYS SEKUNDEN	USratio (base)
101.tomcatv	3700	USER	12	13,5	734,3	5,0
		SYS		0,8		
102.swim	8600	USER	7	33,7	454,1	18,9
		SYS		0,4		
103.su2cor	1400	USER	10	45,0	645,9	2,2
		SYS		0,9		
104.hydro2d	2400	USER	37	35,7	2256,4	1,1
		SYS		0,7		
107.mgrid	2500	USER	20	12,1	1212,8	2,1
		SYS		0,7		
110.applu	2200	USER	12	28,5	749,3	2,9
		SYS		0,8		
125.turb3d	4100	USER	1	43,2	103,8	39,5
		SYS		0,6		
141.apsi	2100	USER	10	57,2	657,9	3,2
		SYS		0,7		
145.fpppp	9600	USER	13	48,4	828,5	11,6
		SYS		0,1		
146.wave5	3000	USER	7	9,7	430,8	7,0
		SYS		1,1		

Geometrisches Mittel US_fp95: 5,19

SPECfp95/US_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

Flags: OPT = -O
Für 107.mgrid sind erforderlich: EXTRA_LDFLAGS = -Wl,-a,archive

Im folgenden Diagramm werden für jeden Benchmark die einzelnen Ratio-Werte und die Durchschnittswerte graphisch veranschaulicht. Die teilweise erheblichen, durch die hohe Nutzerlast hervorgerufenen Differenzen kommen gut zu Geltung, auch wenn durch die "Ausreißer" 102.swim und 125.turb3d eine gewisse Verzerrung entsteht.



Die zwei Säulen zeigen jeweils die Werte SPECratio (baseline_optimiert),
USratio (baseline_optimiert)

in eben dieser Reihenfolge für jeden der 10 Benchmarks.

Die zwei Querlinien zeigen den Gesamtmittelwert SPECfp95_base und den zusätzlich ermittelten "Grenzwert" US_fp95_base. Beachtenswert sind die großen Differenzen zwischen den offiziellen SPECratios und den inoffiziellen Grenzwerten USratio. Sie spiegeln die hohe Prozessorlast wider.

b) SPECrate_fp95:

Da der Rechner Tantalus nur eine CPU besitzt, ist keine spezielle Durchsatzmessung erforderlich, die Werte für SPECrate_fp95 ergeben sich durch Berechnung aus den Werten der Laufzeitmessungen in Teil a) :
(Für den ergänzenden "Grenzwert" USrate ist an die Tabelle eine zusätzliche Spalte angefügt, die Werte sind jeweils aus der Summe aus USER- und SYS-Zeiten der Laufzeitmessungen berechnet, die hier nicht nochmals einzeln gelistet werden.)

BENCHMARK	REF-ZEIT SEKUNDEN	ANZAHL KOPIEN	ZEIT SEKUNDEN	SPECrate	USrate
101.tomcatv	3700	1	2221,9	15,0	45,3
102.swim	8600	1	1378,4	56,2	170,4
103.su2cor	1400	1	1968,5	6,4	19,5
104.hydro2d	2400	1	6829,4	3,2	9,6
107.mgrid	2500	1	4279,4	5,3	18,6
110.applu	2200	1	2287,4	8,7	26,4
125.turb3d	4100	1	308,9	119,5	355,5
141.apsi	2100	1	2024,7	9,3	28,7
145.fpppp	9600	1	2524,2	34,2	104,3
146.wave5	3000	1	1300,8	20,8	62,7
Geometrisches Mittel	(SPECrate_fp95):			15,1	
	(USrate_fp95):				46,5

SPECrate_fp95/USrate_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

Flags: OPT = -O
Für 107.mgrid sind erforderlich: EXTRA_LDFLAGS = -Wl,-a,archive

Bei Versuchen mit optimierenden Compilerschaltern (+O3) konnten keine relevanten Zeitverbesserungen gemessen werden, deshalb wird hier auf eine Auflistung von Werten für optimierte Messungen verzichtet. Das Linker-Flag -Wl,-a,archive wurde dabei ebenso für alle Benchmarks verwendet, brachte jedoch keine grundsätzliche Laufzeitverbesserung, für 107.mgrid muß es allerdings aus Portabilitätsgründen verwendet werden.
Die Schalter der offiziellen SPEC-Messung konnten wegen des späten vorliegens nicht mehr gestestet werden. Für die Messungen auf dieser Maschine gilt daher bei allen Resultaten SPEC...fp95=SPEC...fp95_base. Das Gleiche trifft für die die USER+SYS-Grenzwerte zu.

3.4.3 Erläuterung und Interpretation der Tantalus-Messungen

Neben Herkules und Samson ist der Tantalus der dritte Compute-Server der TU Chemnitz-Zwickau im Testfeld. Man kann sagen, daß dieser Rechner diese Bezeichnung auch am ehesten verdient hat. Dies ist neben der schnellen CPU vor allem auf die gute Hauptspeicherbestückung (144 MByte) zurückzuführen. Es ist eine starke Nutzerfrequentierung zu verzeichnen. Diese äußert sich zum Teil sogar darin, daß zeitweise das Anmelden von anderen Rechnern aus scheitert, weil keine freien Netzwerkports zur Verfügung stehen. Trotzdem liegen Antwort- und Abarbeitungszeiten außer bei hin und wieder auftretenden Lastspitzen in einem akzeptablen Bereich.
Die Betrachtung der SYS-Zeiten zeigt hier, daß diese sonst primär vom Speichersystem (dazu gehört prinzipiell auch der Cache) beeinflußt werden. Sie liegen im Bereich um 1 Sekunde (auch für 110.applu !) und zeigen damit, was in diesem Bereich möglich ist, und daß bei den anderen Maschinen eindeutige Defizite vorhanden sind. Die Annahmen bei der Definition der Grenzwerte auf der Basis der USER- und SYS-Zeiten (Abschnitt 2.2.) werden hier bestätigt.
Die relativ große Spanne zwischen SPECfp95 und US_fp95 (etwa Faktor 3) bzw. zwischen den einzelnen SPECratio und USratio sind auf die hohen Nutzer- und damit Prozeßzahlen zurückzuführen, die sich die verfügbare Rechenzeit teilen müssen. Die schlechtesten Messungen lieferten maximal die doppelte REAL-Zeit und ergaben damit Unterschiede bis zu Faktor 6 zwischen SPECfp95 und Us_fp95.
Die Maschine HP Apollo Serie 700 Modell 735/125 ist die einzige im Testfeld, für die CFP95-Resultate von SPEC zur Verfügung stehen, und zwar in [3], Seite 53 (CFP95) und Seite 110 (CFP95rate). Daher ist an dieser Stelle ein direkter Vergleich möglich. Die bei HP ermittelten Werte sind:

SPECfp95 =SPECfp_base95 =4.55
SPECfp_rate95 =SPECfp_rate_base95 =40,9.

Für diese Gesamtwerte läßt sich jetzt feststellen, daß die hier unter Alltagsbedingungen ermittelten Werte für SPECfp95 und SPECrate_fp95 erheblich schlechter sind, als die bei HP unter Laborbedingungen gemessenen. Beim Vergleich der USER/SYS-Zeiten für die hier gemachten Messungen und der daraus ermittelten Grenzwerte US_fp95=5,19 und USrate_fp95=46,5 mit obigen offiziellen SPEC-Resultaten zeigt sich, daß die Herstellerwerte knapp unterhalb dieser "Grenzwerte" liegen. Auch das bestätigt die Theorie, die der Einführung der (inoffiziellen) Grenzwerte im Abschnitt 2.2. dieser Arbeit zugrunde liegt. Danach sind die verbleibenden Differenzen auf die Wartezeiten für Ein- und Ausgabeoperationen zurückzuführen .
Der Ehrlichkeit halber muß aber erwähnt werden, daß diese Aussage nur bedingt prüfbar ist. So läßt sich bei der Betrachtung der Laufzeiten bzw. der SPECratios von HP und der hier ermittelten Werte USratio auch ablesen, daß die Laufzeiten für einige der Benchmarks in der offiziellen Version gegenüber der für diese Arbeit verwendeten Pre-Release noch verändert wurden. Bei dem auf allen Maschinen einen "Ausreißer" darstellenden 125.turb3d scheint man die Laufzeit durch Quellcodeveränderung einfach verzehnfacht zu haben. Bei einigen anderen (z.B. 102.swim, 104.hydro2d) fielen diese Änderungen moderater aus, wobei bei einigen die Laufzeit auch verkürzt wurde. Dadurch und durch die Verwendung des geometrischen Mittels fallen diese Änderungen bei den Durchschnittswerten nicht so sehr ins Gewicht. Bei einigen Benchmarks (101.tomcatv, 103.su2cor, 110.applu, 145.fpppp) stimmen die offiziellen SPECratio und die hier ermittelten USratio fast überein.
Trotz der vorliegenden 95er Vergleichswerte soll auch der Vergleich mit den offiziellen SPEC-Resultaten mit der CFP92-Suite gezogen werden. Diese sind über den SPEC-WWW-Server [2] oder direkt über folgende Adresse abrufbar:

SPECfp92: http://performance.netlib.org/performance/html/spec.hp735125.cfp92.6_94.notes.html#
{Hewlett-Packard HP 9000 Model 735/125 125MHz PA-RISC 7150}
SPECbase_fp92=183,2

Auf diesem Server finden sich auch SPECfp92-Resultate für die neue Referenzmaschine bei SPECfp95 (SPARCstation 10/40), so daß man die Ergebnisse zumindest in Relation setzen kann:

SPECfp92: (http://performance.netlib.org/performance/html/spec.sunf41.cfp92.6_93.notes.html#
{Sun Microsystems Inc. SPARCstation 10 Model 40 40MHz TMS390Z50})
Geometric Mean (SPECfp92): 60.2

Für diesen Fall liegt das Verhältnis bei 3, also etwa ein Viertel niedriger als SPECfp95 von HP und noch etwas mehr unter dem in den Versuchen ermittelten US_fp95. Dies zeigt, daß der Vergleich der 92er und 95er Werte nur bedingt möglich ist, aber zumindest eine Aussage zuläßt, ob die gemessenen Werte in einer realistischen Größenordnung liegen, so wie dies für die Samson-Messungen gemacht wurde. Auch dort lagen die US_fp95_base-Werte etwa ein Viertel über dem für SS 20/612 und SS10/40 ermittelten Relationsfaktor.
Durchsatzwerte für die 92er und die 95er Suite sind nicht vergleichbar, da sich bei deren Berechnung Normalisierungsfaktoren und Bezugszeitraum unterscheiden.

3.5. DEC 3000-300 AXP (decency.hrz.tu-chemnitz.de)

3.5.1. Gerätekonfiguration

Hardware:

Modell:	DEC 3000-300 AXP
Hersteller:	Digital Equipment Corp.
CPU:	21064 125 MHz
Anzahl CPU's:	1
FPU:	Integriert
Primärcache:	8KB(I) +8KB(D) on chip
Sekundärcache:	256KB(I+D) off chip
Hauptspeicher:	32 MB
Lokale Speicher:	Harddisk 1,9 GB (intern)

Software:

Betriebssystem:	Digital UNIX 3.2 Rel.214 (ehem. DEC OSF/1)
Compiler:	DEC FORTRAN 77 Compiler
	DEC C Compiler
Filesystem:	NFS
Systemstatus:	Multi-User (Tests auch unter echten Multi-User-Bedingungen gelaufen)

weitere Daten:

IP-Name:	decency.hrz.tu-chemnitz.de
IP-Nummer:	134.109.200.68 Ethernet
Standort:	Straße der Nationen
	Böttcherbau, Raum 071

Diese Daten stammen direkt aus dem Universitätsrechenzentrum beziehungsweise aus den Veröffentlichungen des URZ im WWW .
(Adresse: http://www.tu-chemnitz.de/home/ghe/MASCHINEN/dec3000)

3.5.2. Resultate der Messungen

a) SPECfp95(_base) :

Die folgenden Ergebnisse wurden entsprechend der SPEC-Regeln (SPEC-Run-Rules [7]) ermittelt. Es werden die tatsächlichen Laufzeiten der einzelnen Benchmarks betrachtet. Die Auswertung orientiert sich an den SPEC-Regeln zur Veröffentlichung offizieller Ergebnisse (SPEC-Reporting-Rules [8]).

BENCHMARK	SPEC-REFERENZ-ZEIT SEKUNDEN	LAUFZEIT SEKUNDEN	SPECratio (base)
101.tomcatv	3700	1512,0	2,4
102.swim	8600	1433,6	6,0
103.su2cor	1400	2293,6	0,6
104.hydro2d	2400	4480,2	0,5
107.mgrid	2500	1223,1 *	2,0 *
110.applu	2200	30223,8	0,1
125.turb3d	4100	691,7	5,9
141.apsi	2100	583,9	3,6
145.fpppp	9600	2502,6	3,8
146.wave5	3000	3758,2	0,8

Geometrisches Mittel (SPECfp95_base): 1,50* (Baseline-optimierte Messung)

Zusätzlich werden nachfolgende Zeiten betrachtet, die bei jedem Testlauf quasi als Nebenprodukt ermittelt werden. In offiziellen SPEC-Veröffentlichungen sind diese Daten nicht enthalten, sie stellen nur eine Interpretationshilfe für die speziellen Anwendungsbedingungen in diesem Fall dar. Es handelt sich dabei um die USER- und SYS-Zeiten, also die Zeit, in der ein Benchmark tatsächlich einer CPU zugeteilt war.

BENCHMARK	SPEC-REFERENZ-ZEIT	GEMESSENE ZEITEN STD MIN	SEK	SUMME USER+SYS SEKUNDEN	USratio (base)
101.tomcatv	3700	USER	1440,7	1447,9	2,6
		SYS	7,2		
102.swim	8600	USER	1421,1	1423,6	6,0
		SYS	2,5		
103.su2cor	1400	USER	1301,6	1368,6	1,0
		SYS	67,0		
104.hydro2d	2400	USER	4428,2	4432,9	0,5
		SYS	4,7		
107.mgrid	2500	USER	1205,4	1209,1 *	2,1 *
		SYS	3,7		
110.applu	2200	USER	1496,0	3278,2	0,7
		SYS	1782,2		
125.turb3d	4100	USER	190,0	214,5	19,1
		SYS	24,5		
141.apsi	2100	USER	568,2	574,4	3,7
		SYS	6,2		
145.fpppp	9600	USER	2484,9	2487,8	3,9
		SYS	2,9		
146.wave5	3000	USER	1529,2	1717,9	1,7
		SYS	188,7		

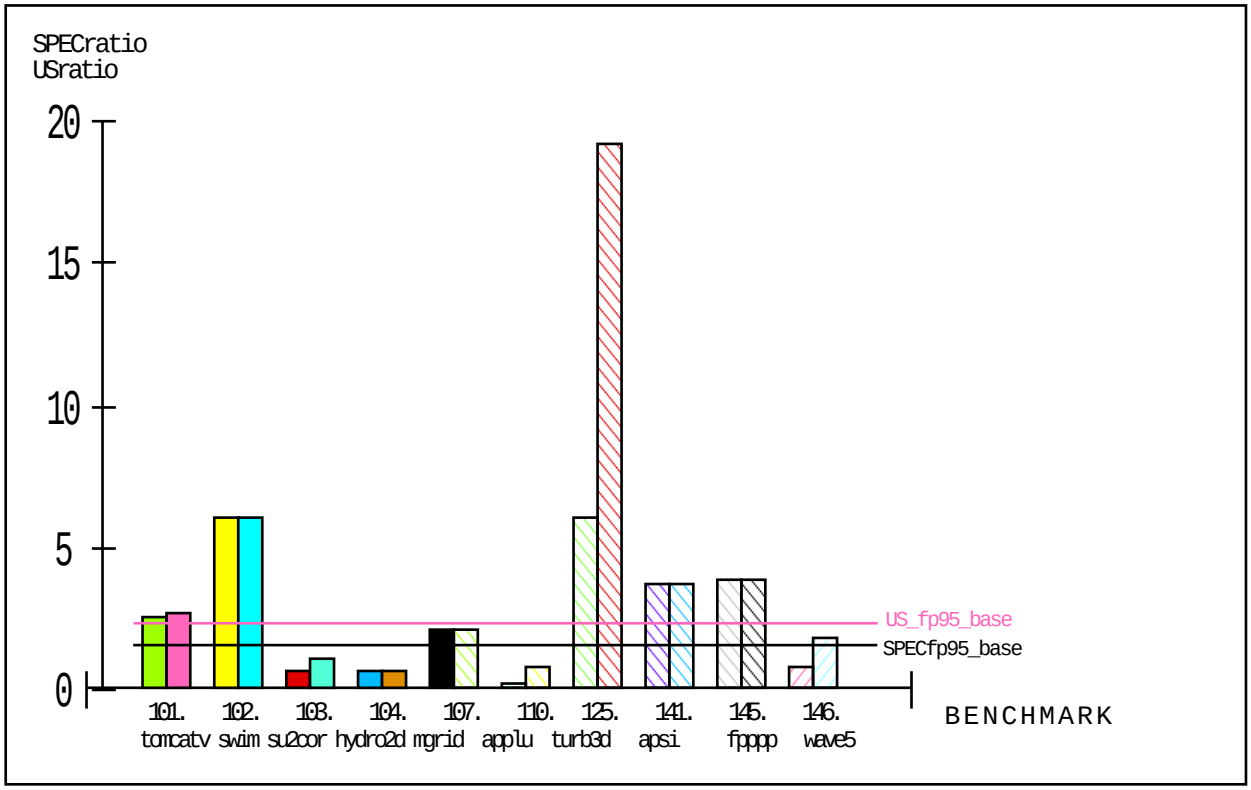
Geometrisches Mittel US_fp95_base: 2,36* (Baseline-optimierte Messung)

SPECfp95/US_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Der Benchmark läuft zum Ende durch, das Vergleichsprogramm spiff aus den SPEC-Tools findet jedoch Differenzen zwischen den vom Benchmark berechneten Resultaten und den zugehörigen SPEC-Referenzwerten. Der Vergleich bricht wegen Speichermangels ab. Ein alternatives Referenzfile für DEC existiert noch nicht, die Optimierungsflags beseitigen den Fehler nicht.

Flags:			
	OPT	=	-O
	EXTRA_FFLAGS	=	-fast -non_shared -om
	EXTRA_LDFLAGS	=	-fast -non_shared -om

Im folgenden Diagramm werden für jeden Benchmark die einzelnen Ratio-Werte graphisch veranschaulicht, sowie die Durchschnittswerte gezeigt. Weit auseinanderliegende SPECratio und US-ratio zeigen den Mangel an einer Ressource, bei der geringen Last und den hohen SYS-Zeiten ist das hier der Hauptspeicher, für den entsprechenden Benchmark an.



Die zwei Säulen zeigen jeweils die Werte SPECratio (baseline_optimiert),
USratio (baseline_optimiert)

in eben dieser Reihenfolge für jeden der 10 Benchmarks.

Die zwei Querlinien zeigen den Gesamtmittelwert SPECfp95_base und den zusätzlich ermittelten "Grenzwert" US_fp95_base.

b) SPECrate_fp95(_base):

Da der Rechner Decency nur eine CPU besitzt, ist keine spezielle Durchsatzmessung erforderlich, die Werte für SPECrate_fp95 ergeben sich durch Berechnung aus den Werten der Laufzeitmessungen in Teil a) :
(Für den ergänzenden "Grenzwert" USrate ist an die Tabelle eine zusätzliche Spalte angefügt, die Werte sind jeweils aus der Summe aus USER- und SYS-Zeiten der Laufzeitmessungen berechnet, die hier nicht nochmals einzeln gelistet werden.)

BENCHMARK	REF-ZEIT SEKUNDEN	ANZAHL KOPIEN	ZEIT SEKUNDEN	SPECrate	USrate
101.tomcatv	3700	1	1512,0	22,0	23,0
102.swim	8600	1	1433,6	54,0	54,4
103.su2cor	1400	1	2293,6	5,5	9,2
104.hydro2d	2400	1	4480,2	4,8	4,9
107.mgrid	2500	1	1223,1	* 18,4 *	18,6 *
110.applu	2200	1	30223,8	0,7	6,0
125.turb3d	4100	1	691,7	53,3	172,0
141.apsi	2100	1	583,9	32,4	32,9
145.fpppp	9600	1	2502,6	34,5	34,7
146.wave5	3000	1	3758,2	7,2	15,7
Geometrisches Mittel	(SPECrate_fp95_base):			13,2*	
	(USrate_fp95_base):				21,3*

SPECrate_fp95/USrate_fp95 Fehler/Tuning-Parameter/Bemerkungen/Änderungen:

* Der Benchmark läuft zum Ende durch, das Vergleichsprogramm spiff aus den SPEC-Tools findet jedoch Differenzen zwischen den vom Benchmark berechneten Resultaten und den zugehörigen SPEC-Referenzwerten. Der Vergleich bricht wegen Speichermangels ab. Ein alternatives Referenzfile für DEC existiert nicht, die Optimierungsflags beseitigen den Fehler nicht.

Flags:	OPT	=	-O
	EXTRA_FFLAGS	=	-fast -non_shared -om
	EXTRA_LDFLAGS	=	-fast -non_shared -om

Es wurden auch (nichtoptimierte) Standardmessungen vorgenommen (nur -O ---> default-Optimierung mit Stufe O4). Dabei traten auch bei 103.su2cor Differenzen zu den SPEC-Referenzresultaten auf. Die USER-Zeiten lagen nur 1..3% über den hier angegebenen Resultaten der Baseline-optimierten Messung, daher wurde bei diesem Report auf die Werte der Standardmessung verzichtet.

3.5.3 Erläuterung und Interpretation der Decency-Messungen

Für diesen Rechner treffen hinsichtlich der Belastung in etwa dieselben Aussagen zu, wie für die bereits betrachtete SGI-Maschine silly. Die Frequentierung durch entfernte Nutzer ist (speziell in den Nachtstunden) relativ gering.
Allerdings wird auch allein mit den Benchmarks die Leistungsgrenze des Rechners sehr schnell erreicht. So kam es während der am Tage gemachten Test mehrfach zu Beschwerden von anderen Nutzern, da deren (wenige) Anwendungen teilweise zum Stehen kamen. Ursache ist die geringe Speicherausstattung mit 32 MByte, die bei derartiger Belastung die von Hause aus guten Rechenleistungen des Alpha-Prozessors, ablesbar an den reinen USER-Zeiten, nicht zur Geltung kommen läßt. Es bewahrheiten sich die Aussagen von Reinhold Weicker [1] und jene aus [3], daß für eine realistische Anwendung der Benchmarks 64 MByte Hauptspeicher schon vorhanden sein sollten.
Der Einsatz der virtuellen Speicherverwaltung (Paging) läßt sich an den hohen SYS-Zeiten bei 110.applu, 146.wave5, 103.su2cor und 125.turb3d ablesen, wie es speziell bei den beiden ersten auch schon auf dem Herkules mit 32 MByte Hauptspeicher zu verzeichnen war. Dieses Paging bewirkt sofort eine erhebliche Differenz zwischen SPECratio und dem theoretischen Grenzwert USratio. Bei Messungen mit höherer Last (am Tage) zeigten sich Paging-Effekte auch bei 101.tomcatv und 141.apsi, was auf dem Herkules eigentlich nie der Fall war. Bei solchen Messungen mit höherer Last wurden bis zu 5-fache Werte der hier dokumentierten REAL-Zeiten gemessen. Der Rechner Decency wird also mit einer höheren Belastung sichtbar schlechter fertig, als die anderen Testrechner. Trotz des schnellen Prozessors fällt die Leistungsfähigkeit mit steigender Belastung schnell ab. Obwohl die "normale" Last auf dieser Maschine auf einem niedrigeren Level liegt, als auf den Vergleichsrechnern, werden die Benchmarks besonders stark gebremst. Es zeigt sich, daß ein schneller Prozessor nicht alles ist, hier liegt der Flaschenhals im Umfeld. Eine exakte Eingrenzung der Ursache ist unter den vorhandenen Meßbedingungen nicht möglich. Zwei Interpretationen des Abfallens gegenüber dem Rechner Herkules sind denkbar. Zum einen könnte man auf differenzierte Strategien bei der Auswahl der auszulagernden Seiten beim Paging in den Betriebssystem-Implementierungen der einzelnen Hersteller schließen. So lagen auf der DEC-Maschine die SYS-Zeiten stets höher als auf der SUN-Maschine, obwohl die eigentliche Prozeßlast durchschnittlich geringer war und der Prozessor nach der reinen USER-Zeit eigentlich schneller ist. Der zweite mögliche Grund könnte die unterschiedliche Cache-Konfiguration der beiden Rechner sein, nach der die SUN-Maschine klar bevorteilt sein müßte. Jedoch kann man bei hohen Page-Miss-Raten auch eine hohe Anzahl von Cache-Misses annehmen.
Beide gemachten Annahmen sind aber Hypothesen, eine genaue Analyse kann nur mit Meßreihen erfolgen, bei denen die Beeinflussung jeweils auf eine Randbedingung beschränkt wird, die anderen alle gleich sind, was schon bei der Cache-Konfiguration ziemlich schwierig sein dürfte.
Eine Veröffentlichung offizieller Meßergebnisse für eine gleichartige Maschine liegt von SPEC für CFP95 noch nicht vor. Somit bleibt wieder nur der Vergleich der Größenordnung mittels Umrechnung von CFP92-Ergebnissen. Eine Veröffentlichung von CFP92-Resultaten für DEC 3000 Modell 300 LX (allerdings mit 64MByte Hauptspeicher) findet man auf dem SPEC-WWW-Server [2] unter der Adresse

http://perfomance.netlib.org/performance/html/spec.dec3-3lx.cfp92.6_94.notes.html#
{Digital Equipment Corporation DEC 3000 Model 300LX 125MHz 21064}

Der dort ermittelte Durchschnittswert SPECbase_fp92 liegt bei 73,1.
Verglichen mit den mit der CFP92-Suite gemessenen Ergebnissen für die neue SPEC -Referenzmaschine SPARCstation 10/40, veröffentlicht unter der Adresse

http://perfomance.netlib.org/performance/html/spec.sunf41.cfp92.6_93.notes.html#
{Sun Microsystems Inc. SPARCstation 10 Model 40 40MHz TMS390Z50}
Geometric Mean (SPECfp92): 60.2

ergibt sich als Verhältnis der Faktor 1,2. Auf die nur bedingt vorhandene Vergleichbarkeit von CFP95- und CFP92-Messungen und die Problematik der noch nicht abgeschlossenen Benchmark-Entwicklung wurde bereits mehrfach hingewiesen. Auch hier kommt durch die relativ kleinen SPECratio-Werte der "Ausreißer" 125.turb3d stark zur Geltung, weil er die Durchschnittswerte nach oben verfälscht. Ohne 125.turb3d ergeben sich Durchschnittswerte von SPECfp95_base=1,3 und US_fp95_base=1,8. Damit ergibt sich beim Unterschied zwischen 92er und 95er Werten in etwa die gleiche Größenordnung wie beim gut vergleichbaren Tantalus, womit gezeigt wird, daß die protokollierte Messung realistisch ist.

Die verwendeten Compilerschalter entstammen der oben angegebenen Veröffentlichung von 92er SPEC-Resultaten auf dem SPEC-WWW-Server.

4. Gesamtauswertung und Resümee

Der Test der Benchmarksuite CFP95 (Pre-Release CFP95.29) auf den 5 ausgewählten Plattformen hat wie erwartet gezeigt, daß die unter Laborbedingungen bei SPEC-Mitgliedsfirmen gemessenen Leistungswerte bei Messungen unter praktischen Einsatzbedingungen erheblich abweichen können, daß man also nicht von hohen SPEC-Werten in den Verkaufsprospekten automatisch auf hohe Benchmarkleistungen im Alltagsbetrieb schließen sollte. Auch ohne das Vorliegen von offiziellen Vergleichswerten (außer für HP-PA) muß man bei den Versuchen unter Alltagsbedingungen zu diesem Ergebnis kommen, da schon bei unterschiedlichen Lastbedingungen die Resultate der Messungen um den Faktor 2 bis 5 differieren können.
Die neuesten Veröffentlichungen in [2], [3] aus Anlaß der Herausgabe der neuen CPU-Benchmarksuites CFP95 und CINT95 enthalten auch mehrfach den Hinweis, daß die Ergebnisse nicht nur von der CPU, sondern auch von den Komponenten Speicherhierarchie und Compiler beeinflußt werden, trotzdem aber keinen Leistungsindex für ein komplettes System darstellen, da I/O-System, Graphik- und Netzwerkleistungsfähigkeit unberücksichtigt bleiben.
Die Hersteller messen die Leistungswerte für ihre neuen Produkt unter möglichst optimalen Bedingungen. Bei den Marktverhältnissen auf dem Hardwaresektor ist dies nur allzu verständlich. Kein Hersteller kann es sich leisten, beim Verkaufsargument "SPEC-Leistung" auch nur Bruchteile von Leistungspunkten zu verschenken. Es ist jedoch nicht ganz verständlich, daß SPEC selbst keine Tools liefert, die es erlauben, die unter Praxisbedingungen bei Anwendern erreichten Meßwerte zu relativieren und zumindest eine grobe Ursachenanalyse zu ermöglichen. Immerhin verkauft SPEC die Benchmarksuites an jeden interessierten Rechneranwender oder -produzenten. Diese Käufer erwerben die Benchmarks doch aber nicht, um ihre Maschinen zu isolieren und durch Rekonstruktion der SPEC-Versuchsbedingungen die Ergebnisse der Herstellermessungen zu wiederholen. Vielmehr ist das Ziel darin zu sehen, die theoretisch möglichen Leistungswerte auch im Praxisbetrieb umzusetzen und das Gesamtsystem in diese Richtung zu optimieren. Dazu ist die Verwendung der SPEC-CPU-Benchmarks jedoch ein wenig taugliches Mittel, zumindest wenn man nur die offiziell zu ermittelnden und zu veröffentlichenden SPEC-Resultate betrachtet.
Die für diese Arbeit gemachten Versuche haben gezeigt, daß bei näherem Betrachten des eigentlichen Meßverfahrens sich schon ein paar Aussagen über die Beeinflußbarkeit der Benchmarkresultate gewinnen lassen. Das relativ primitiv anmutende Zeitmeßverfahren mit dem UNIX-Kommando time liefert hierfür den Ansatz. Für die offizielle Ergebnispublizierung wird nur die Gesamtlaufzeit eines Benchmarks betrachtet, also die Differenz zwischen End- und Startzeit des Programms. Dies ist der vom time-Kommando gelieferte "REAL"-Wert. time liefert jedoch zwei weitere Zeitwerte "USER", also die CPU-Zuteilungszeit für den gemessenen Nutzerprozeß, und "SYS" als die Zeit, die während der Messung direkt in Systemoperationen benötigt wurde. Diese Zeiten werden von SPEC für keinerlei Auswertungen verwendet, in offiziellen SPEC-Dokumenten werden sie nicht einmal erwähnt.
Das Feld von 5 Testrechnern mit unterschiedlichen Systemen lieferte ein relativ breites Spektrum von unterschiedlichen Konfigurationen und Einsatzbedingungen. So kann man die Rechner beispielsweise folgendermaßen einordnen:

- Multiprozessor-Computeserver mit großem Speicher und hoher Nutzerlast, 2 Prozessoren
  ---> Rechner SAMSON
- Singleprozessor-Computeserver mit großem Speicher, hoher Nutzerlast, sehr schnelle CPU
  ---> Rechner TANTALUS
- Singleprozessor-Computeserver mit wenig Speicher , aber hoher Last durch Nutzerprozesse
  ---> Rechner HERKULES
- Singleprozessorrechner mit typischer Workstationbenutzung, also relativ geringe Last bei nahezu ausreichend Speicher
  ---> Rechner SILLY
- Singleprozessorrechner mit typischer Workstationbenutzung, sehr schnelle CPU, aber (zu) wenig Hauptspeicher
  ---> Rechner DECENCY

Betrachtet man für diese Rechner die gemessenen (inoffiziellen) USER- und SYS-Zeiten im Zusammenhang mit diesem Rechnerumfeld, so kann man schon einige Schlüsse über die Beeinflussung der Ergebnisse durch einzelne Bedingungen im Umfeld ziehen. Ein Teil dieser Effekte wurde bereits bei den Auswertungen der Einzelversuche kommentiert und am Beispiel gezeigt, diese Einzelfakten sollen im folgenden zusammengefaßt

werden. Es muß nochmals ausdrücklich darauf hingewiesen werden, daß diese Aussagen zu einem großen Teil nur Hypothesen sind, die aus einer Reihe von Messungen mit unterschiedlichen Bedingungen über einen längeren Zeitraum hinweg abgeleitet wurden. Die beschriebenen Effekte traten über dieses gesamte Meßspektrum hinweg auf, für die Interpretation wurden die eben dargestellten Annahmen über die typischen Lasten auf den einzelnen Rechnern verwendet. Natürlich traten über den Gesamtzeitraum hinweg Ausnahmen in der Belastung nach unten wie nach oben auf, im Durchschnitt treffen diese Umfeldabschätzungen jedoch zu. Auf die Abweichungen und die Ergebnisstreuung wurde bei den Einzelauswertungen hingewiesen. Eine genaue Analyse kann nur mit Messungen erfolgen, bei denen man den Einfluß der Randbedingungen auf jeweils eine Einflußgröße reduziert, was praktisch bedeutet, mit gleichen Lasten und Konfigurationen zu messen. Dies war im Rahmen dieser Arbeit nicht möglich, es konnten nur Abschätzungen gewonnen werden, die den Grundstock für künftigen Arbeiten legen sollen.
Zur Illustration der auftretenden Effekte werden exemplarisch die folgenden Meßwerte für den Benchmark 110.applu auf den verschiedenen Maschinen verwendet, so wie dies schon bei der Einzelauswertung der Herkules-Messungen praktiziert wurde. Dieser Benchmark ist besonders speicherintensiv, wodurch sich speziell die Pagingeffekte gut zeigen lassen.

```
HERKULES:
/a/zuse-f/home/urz/fs22/c/cmun/Projekt/HERK/benchspec/110.applu/time.out
    41114.9 real       2650.8 user       1792.4 sys
HERK: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)

DECENCY:
/a/zuse-f/home/urz/fs22/c/cmun/Projekt/DEC/benchspec/110.applu/time.out
real   30223.8
user    1496.0
sys     1782.2
: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)

SAMSON:
a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/benchspec/110.applu/time.out
real     41:11.7
user     31:31.5
sys          2.8
CFP95.29: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)

TANTALUS:
/a/zuse-f/home/urz/fs22/c/cmun/Projekt/TANT/benchspec/110.applu/time.out
real    38:07.4
user    12:28.5
sys         0.8
: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)

SILLY:
/a/zuse-f/home/urz/fs22/c/cmun/Projekt/SILLY/benchspec/110.applu/time.out
real      29:00.49
user      28:24.51
sys          6.97
SILLY: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)
```

Die Summe aus USER- und SYS-Zeit ist die eigentliche Abarbeitungszeit für einen Benchmark, daher wird dieser Wert als "Grenzwert US(rate)_fp95" für die CPU-Leistungsfähigkeit der entsprechenden Maschine betrachtet. Es wäre auch noch denkbar, nur die USER-Zeit zu verwenden, damit wird der Einfluß des Speichersystems nicht mehr mit betrachtet.
Die Benchmarks enthalten fast keine Ein- und Ausgabeoperationen, im Optimalfall liegen also tatsächliche Laufzeit (REAL) und CPU-Zeit (USER+SYS) eng beieinander, weil nicht auf langsamere I/O-Geräte gewartet werden muß. (Siehe SILLY, man kann dies aber auch bei nicht so speicherintensiven Benchmarks bei den Decency-Messungen mit wenig Last ablesen.)
Die Systemzeit ist im Idealfall sehr gering, da die Benchmarks zum Zwecke der einfachen Portierbarkeit ohne Verwendung von systemnahen Funktionen sehr einfach implementiert sind. Sehr gering bedeutet, daß sie unter einer Sekunde (TANTALUS) bzw. im Bereich weniger Sekunden (SILLY, SAMSON) liegen. Ein erhebliches Ansteigen dieser SYS-Zeit ist also Zeichen dafür, daß das Betriebssystem in seiner Funktion zur Koordinierung von Ressourcen gefordert wird. Bei den Benchmarks ist dies primär das Betriebsmittel Hauptspeicher. Ein Ansteigen der SYS-Zeit in den Minutenbereich läßt nur den Schluß zu, daß dies auf die Aktivierung der

virtuellen Speicherverwaltung (Paging) zurückzuführen ist, bei der die Sekundärspeicher (Harddisk) den "Flaschenhals" bei der Geschwindigkeit darstellen. Neben der hohen SYS-Zeit ist ein starkes Ansteigen der REAL-Zeit zu verzeichnen, da zur Systemzeit für das Paging noch die Wartezeit für das Zurückladen ausgelagerter Speicherseiten in den Hauptspeicher kommt. Einzelne Unterschiede bei verschiedenen Rechnern mit vergleichbaren Speichergrößen können außer auf unterschiedliche Lastverhältnisse auch auf unterschiedliche Auslagerungsstrategien beim Paging in den einzelnen Betriebssystemen oder die unterschiedlichen Cache-Konfigurationen zurückzuführen sein. Dies tritt beim Vergleich zwischen HERKULES und DECENCY auf, kann aber in dieser Arbeit nicht näher untersucht werden, obwohl es interessant wäre zu ermitteln, warum die SYS-Zeit trotz des eigentlich schnelleren Prozessors und der allgemein geringeren Last beim DECENCY auf demselben Niveau liegt, wie beim HERKULES, und warum der DECENCY beim Ansteigen der Last so schnell in seiner Leistung abfällt.
Nimmt die Anzahl der Nutzerprozesse auf einer Maschine zu, so steigt logischerweise die Gesamtlaufzeit (REAL), da sich mehr Bewerber die begrenzte Ressource "CPU-Zeit" teilen müssen. Eine große Differenz zwischen der REAL-Zeit und der (USER+SYS)-Zeit ist bei Messungen ohne Paging allein auf die Prozessorlast während des Programmlaufes zurückzuführen (SAMSON und TANTALUS). Für die Bewertung eines Systems mit den vorliegenden CPU-Benchmarks unter Alltagsbedingungen ist es folglich unbedingt erforderlich, vor den Messungen eine Analyse der typischen Lastverhältnisse auf dieser Maschine zu machen. Dies sollte anhand der im Universitätsrechenzentrum erstellten Statistiken möglich sein.
An dieser Stelle sei auf zwei weitere Einflußfaktoren hingewiesen, deren Einfluß mit den hier gemachten Messungen nicht sicher ermittelt werden konnte, da eventuelle Effekte durch jene der Speicher- und Lastproblematik verdeckt werden können. Damit ist der Einfluß der Caches und des Netzwerk-Filesystems gemeint. Die Datencaches dürften einen nur geringen Einfluß haben, da der Fall des Cache-Miss wohl der Standardfall sein dürfte, wenn sogar Paging während der Abarbeitung erforderlich ist. Die beiden analysierten Benchmarks weisen ein bezüglich der Speicherpositionen ausgewogenes Zugriffsverhalten auf.
Etwas anders verhält sich die Sache beim Befehlscache. Speziell bei den Durchsatzmessungen mit mehreren Programmkopien kann dieser die Abarbeitung für die zweite, dritte ... Kopie beschleunigen, da der Code nur einmal geladen werden muß und dann für die Kopien (vielleicht) im Instruction-Cache bereitsteht. Zur Illustration seien hierzu noch die Werte einer Messung mit zwei 110.applu-Kopien auf der Zweiprozessormaschine SAMSON genannt:

```
a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/benchspec/110.applu/time.out
real    1:28:25.6
user      58:02.0
sys           6.8
CFP95.29: BENCHMARK COMPLETED SUCCESSFULLY (using ref reference)
```

Die REAL-Zeit ist mehr als doppelt so hoch wie bei der vorherigen Einzelmessung, dies liegt an der anliegenden Last. Dennoch erreicht die lastunabhängige USER-Zeit nicht den doppelten Wert der Einzelmessung, was eben auf den Einfluß des Instruction-Caches zurückzuführen ist. Generiert man zusätzlich weitere Last (zusätzliche Prozesse), so schwächt sich dieser Effekt immer weiter ab und verschwindet, da bei mehr (verschiedenen) Prozessen für jeden einzelnen die Cache-Misses zunehmen.
Beim Laden des Codes (und der benötigten Daten) kommt man auch zum Einfluß des Netzwerk-Filesystems. Die Benchmarks waren während der Versuche auf einem entfernten Rechner (Fileserver zuse.hrz) gespeichert. Die Ladezeit hängt also nicht nur von Zugriffszeit und Übertragungsrate der Harddisks, sondern auch vom dazwischenliegenden Übertragungsmedium ab. Bei geringer Last dürfte über das Netz keine wesentlich längere Ladezeit verzeichnet werden. Da die Last im Netz aber (fast) immer sehr hoch ist (auch oder gerade in den Nachtstunden), können einige Sekunden Verzögerung schon möglich sein. Diese ist jedoch an den SYS/USER/REAL-Zeiten nicht direkt ablesbar, wenn man die anderen Bedingungen nicht konkret auf diese spezielle Untersuchung einrichtet. Für die Bewertung der Netzwerkleistungsfähigkeit steht von SPEC ein spezieller Benchmark (SFS) zur Verfügung.
Noch erwähnt werden muß, daß sich alle diese Effekte für die Benchmarks auch auf den gut ausgestatteten Maschinen (TANTALUS, SAMSON) provozieren lassen, wenn man die nötige Speicher- bzw. Prozeßlast erzeugt. Am einfachsten gelingt dies durch den Start mehrerer Benchmarks gleichzeitig. Speziell die Paging-Effekte treten dann auf, allerdings nicht mit einem derartigen Anstieg der Gesamtzeiten wie bei HERKULES oder DECENCY, da sich die Paging-Aktivitäten auf das Zurückladen der Speicherseiten beim Prozeßwechsel beschränken. Für den einzelnen aktiven Prozeß ist dann genügend Speicher da, während auf den "kleineren" Maschinen zu wenig Gesamtspeicher vorhanden ist. Es gibt also immer eine kritische Grenze, die ein Anwender für seine typischen Anwendungen kennen und bei seiner Hardware-Ausstattung nicht unterschreiten sollte. Auch auf den Computeservern könnte man durch immer weitere Laststeigerung den Effekt des plötzlichen

Leistungsabfalls erreichen, im Interesse des öffentlichen Rechenbetriebes wurde darauf verzichtet.
Die eben gemachten Analysen liefern nicht mehr als einen Ansatz für die Betrachtung und Einbeziehung der Nebenbedingungen in die Messungen und Auswertungen. Um genauere Informationen zu gewinnen, muß man tatsächlich eine Maschine isolieren und dann mit Hilfe von Lastmodellen bezüglich jeder Komponente einzeln untersuchen. Die folgenden Schritte könnten das Gerüst für eine weitere Arbeit darstellen, die Aussagen über den konkreten Einfluß der einzelnen Ressourcen liefert.

- Isolierung eines Rechners und Wiederholung der Messung entsprechend einer SPEC-Publikation
- Wiederholung dieses Test mit auf entferntem Rechner befindlichem Programmcode und E/A-Ziel, ohne jedoch andere Nutzer auf dem zu messenden Rechner und dem Netzwerk zuzulassen
--->Aussage über Einfluß des unbelasteten Netzes
-Wiederholung des gleichen Tests mit einer typischen Auslastung des Netzwerkes
---> Aussage über Netzwerkeinfluß im Alltagsbetrieb
- verschiedene Messungen mit verschiedenen Cache- und Speicherausstattungen auf dem isolierten Rechner (falls dies konfigurierbar ist), um konkreten Einfluß und die kritischen Grenzen für einzelnen Benchmarks zu ermitteln
- verschiedene Messungen mit unterschiedlichen, für verschiedene Anwendungsgebiete charakteristischen Lastmodellen, die den typischen Lastverhältnissen auf zu analysierenden Rechnern (Computeserver, Workstations, Kommunikationsserver) entsprechen. Diese Messung sollte natürlich wieder auf der isolierten Maschine erfolgen. Auf nicht isolierbaren Maschinen muß die Last gemssen und in einem konkreten Wert gefaßt werden, um durch Lastunterschiede differierende Resultate relativieren zu können.

Nach einer solchen Meßreihe kann man dann konkret sagen, wie stark sich der Einfluß von Einzelkomponenten auf das Gesamtergebnis auswirkt. Im Umkehrschluß könnte man für die im Alltagsbetrieb getestete Maschine dann in etwa feststellen, durch welche Systemkomponente das gegenüber dem offiziellen SPEC-Resultat schlechtere Meßergebnis maßgeblich hervorgerufen wird und wie die Maschine und die spezielle Komponente gemäß dem eigenen Anspruch an die Leistung verändert (sprich aufgerüstet) werden muß.
Die Durchführung solcher Meßreihen sollte aber nicht allein dem Anwender aufgebürdet werden, dies sollte auch Aufgabe der in der SPEC engagierten Hersteller sein. Die einfachste Möglichkeit wäre die Definition einiger Standard-Lastmodelle, mit denen die Benchmarks dann zusätzlich gefahren werden müssen und deren Ergebnisse in die Veröffentlichungen einfließen. Dies bringt natürlich nur etwas, wenn man zu der Gesamtlaufzeit noch die Einzelzeiten (USER und SYS) in die Betrachtung einbezieht.
Man muß sich jedoch fragen, ob dann nicht langsam "mit Kanonen auf Spatzen geschossen wird". Für die Analyse bestehender Konfigurationen gibt es wohl bessere Werkzeuge als stundenlange Benchmarktests.
Auch eine Auswahl von ca. 3 bis 5 Einzelbenchmarks könnte ausreichend sein. Voraussetzung wäre allerdings die Beseitigung des Informationsdefizits bezüglich der genauen Anforderungen der einzelnen Benchmarks. Bei einer besseren Klassifizierung bezüglich der Anforderungen könnte sich jeder "seine" Klasse von Tests auswählen. Es gilt ein angemessenes Verhältnis von Aufwand und Nutzen zu finden. Betrachtet man die vorherigen Analysen, so ist dies wohl bei tagelangen Meßreihen für den "Normalanwender" nicht mehr gegeben.
Bei SPEC gibt es Bestrebungen, sogenannte heterogene Benchmarks zu entwickeln, bei denen zufällig viele und zufällig lange Benchmarks über einen bestimmten Zeitraum in zufälliger Verteilung laufen, um so einen typischen Rechenbetrieb zu simulieren. Diese Entwicklung steht jedoch erst am Anfang, da viele Probleme noch ungeklärt sind, speziell die Beeinflussung und Festlegung der Testanzahlen und -streuungen, schließlich müssen die Ergebnisse auch repräsentativ und, was nach der SPEC-Philosophie besonders wichtig ist, reproduzierbar werden.

Somit schließt sich der Kreis zum Beginn dieser Arbeit. Die Benchmarkergebnisse SPECfp95, SPECint95, SPECrate_fp95, SPECrate_int95 werden wohl hauptsächlich das bleiben, was ihre Vorgänger waren, nämlich Verkaufsargumente in den Prospekten der Hersteller. Die anderen SPEC-Benchmarkergebnisse SFS, SDM, die nicht einzelne Rechner berücksichtigen, sondern auch die Einflüsse der Multiuserbenutzung und verteilter Filesysteme, werden durch die heterogenen Netzwerkwelten an Bedeutung gewinnen. Dennoch wäre für die Rechneranwender eine Kombination aller dieser Tests mit insgesamt reduziertem Umfang wohl die beste Lösung.
Die zusätzlich ermittelbaren Werte, die auf USER-und SYS-Zeiten basieren, liefern dem Anwender dennoch nützliche Informationen zu den Anfangs genannten Zielen
-Einordnung von älteren Maschinen, die von SPEC-Firmen nicht mehr gemessen werden
-Analyse laufender Systeme zur Defizitermittlung
-Ermittlung des Einflusses von Zusatzsoftware auf die Performance des Gesamtsystems.

An dieser Stelle soll als Abschluß noch kurz auf das Problem der Hardware-Neukaufentscheidungen eingegangen werden, da sich die vorherigen Betrachtungen mit der Analyse laufender Systeme beschäftigten.
Die Versuche haben gezeigt, daß es nicht sinnvoll ist, Kaufentscheidungen nur im Vergleich von SPEC-Werten und Preis für die entsprechenden Maschine zu treffen. Wichtig ist die Eingrenzung des künftigen Aufgabengebietes für den Rechner und die Auswahl der Komponenten.
Für einen Computeserver ist nicht unbedingt erforderlich, daß die höchsten SPECratios erreicht werden, wichtiger sind die Durchsatzleistungen. Es ist also ein Multiprozessorsystem zu bevorzugen, auch wenn die einzelne Anwendung vielleicht etwas länger braucht als auf einer Einzelprozessormaschine mit High-End-Prozessor. An Hauptspeicher und Caches kann theoretisch nie genug vorhanden sein. Praktisch sollten gewisse Grenzen nicht unterschritten werden, die durch Analyse der typischen Nutzer- und Speicherlasten zu ermitteln sind. Im Gegensatz dazu ist der Einsatz von Multiprozessorrechnern als Workstation, die mehr oder weniger im Einzelnutzerbetrieb läuft, als eine Ressourcenverschwendung zu sehen. Letztendlich kommt es auf die Ausgewogenheit aller Komponenten an.
Diese Auswahl kann und darf nicht nur auf der Bewertung einer CPU-Leistungsfähigkeit basieren !

Quellen

[1] Dr. Reinhold Weicker
Kontaktadresse (EMail): weiker.pad@sni.de
Dr. Reinhold Weicker ist verantwortlich für SPEC-Aufgaben bei
Siemens-Nixdorf Informationssysteme AG, D-33094 Paderborn, Germany,
Mitglied im SPEC Steering Committee und
offizieller Ansprechpartner der SPEC für den europäischen Raum

[2] SPEC-WWW-Server;
(URL: http://performance.netlib.org/performance/html/spec.html)
-Hier erfolgt mit Verzögerung die Veröffentlichung von Benchmark-Ergebnissen der SPEC-Mitgliedsfirmen, alle älteren Ergebnisse sind hier verfügbar. -

[3] SPEC Newsletter, Volume 7, Issue 3, September 1995
(SPEC c/o NCGA, Suite 200, 2722 Merrilee Drive, Fairfax, VA 22031-4499)

[4] Weicker, Reinhold P.: Weiterentwicklungen der SPEC-Testsuite; Nicht nur für SPECialisten; iX 2/1994; S.124-130

[5] SPEC Steering Committee: Introduction to SPEC CPU Floating Point Benchmark Suite CFP92; 19. Oktober 1992; S.1-6
(File INTRO im SPEC-Wurzelverzeichnis der Suite CFP95 Vorversion CFP95.29)

[6] SPEC Steering Committee: SPEC CFP92 README File, 19.Oktober 1992; S.1-18
(File README im SPEC-Wurzelverzeichnis der Suite CFP95 Vorversion CFP95.29)
-Allgemeine Hinweise für Installation und Betriebsarten der Benchmarks (CFP92)-

[7] SPEC Steering Committee: SPEC Run Rules for CFP92, 19.Oktober 1992; S. 1-7
(File RUNRULES im SPEC-Wurzelverzeichnis der Suite CFP95 Vorversion CFP95.29)
-Regeln für die korrekte Anwendung der Benchmarksuite CFP92-

[8] SPEC Steering Committee: SPEC Reporting Rules for CFP92, 19.Oktober 1992; S.1-18
(File RPT_RULES im SPEC-Wurzelverzeichnis der Suite CFP95 Vorvers. CFP95.29)
-Regeln für die korrekte Auswertung und Dokumentation von Messungen mit CFP92-

[] Weitere Kurzbeschreibungen zu den einzelnen Benchmarks in den einzelnen Benchmarkverzeichnissen ($SPECHOME/benchspec/$BENCHMARKNAME)

Anlage A Installation - Protokolle und notwendige Änderungen

Teil I - Installation von Tools und Umgebung

Dekomprimieren des Paketes CFP95.29.tar.Z:

```
samson% uncompress CFP95.29.tar.Z
...

samson% tar -xvf  CFP95.29.tar
...
```

----> Verzeichnis wird angelegt und Paket ausgepackt.
Jetzt ist es erforderlich, die Umgebung einzurichten (Suchpfade setzen, Umgebungsvariablen). Dafür steht ein cshrc-File für C-shell-Benutzer oder ein shrc-File für Bourne- oder Korn-Shell-Nutzer zur Verfügung.

```
samson% cd CFP95.29
samson% source cshrc
Setting up your environment.
setenv SPEC [/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29]
Setting DISKDEVS=NONE
The 'bin' directory for SPEC is now being added to your path
path=(. /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bin /uni/global/bin/X11R5 /home/urz/user/c/cmun/bin
/home/urz/user/c/cmun/sun5/bin /usr/local/bin /uni/dept/bin /uni/global/bin /uni/global/opt/SUNwspro/bin /usr/ccs/bin
/usr/afsws/bin /usr/bin /bin /usr/ucb .)
setting TESTRESULTS to /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/tests.results
The make command has been aliased so that the 'make' program will
always execute /bin/sh rather than /bin/csh or /bin/ksh since all of the
makefiles for SPEC were written to /bin/sh for portability.
Checking for a bin directory
Created /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bin
Checking for a /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/tests.results
Created /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/tests.results
setting SPECTMP to /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/tmp
Checking for a tmp directory
Created /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/tmp
samson% rehash
samson%
```

Diese bisherigen Schritte sind für alle Plattformen gleich. Das Setzen der Umgebungsvariablen (source cshrc) ist vor Beginn von Messungen immer erforderlich.
Nachfolgend sind die für die einzelnen Plattformen erforderlichen Änderungen beschrieben, dabei wurden nur diese Änderungen ausgeführt, die erforderlich waren, damit sich die Benchmarks überhaupt übersetzen und Ausführen ließen.
Da die Ergebnisfiles direkt ausgewertet wurden, blieben etwaige Fehler in den Bildschirmausgaben unbeachtet, die durch unterschiedliche Formate von Kommandos und deren Ausgaben (z.B. beim Time-Kommando) oder die Shell-Sprache der einzelnen Plattformen erzeugt werden. Dies ist eher Aufgabe der in der SPEC engagierten Firmen. Die offizielle Version wird mit einer neuen, auf PERL basierenden Umgebung und Auswertung versehen, von der bisher noch nichts zu sehen ist.
Für die Übersetzung auf dem Rechner SAMSON unter dessen Betriebssystem SOLARIS 2.3 ist das vollständige Installationsprotokoll angegeben. Für die anderen 4 Plattformen wird dies auf Auszüge (jeweils mit Fehlermeldungen und die Behebung der Fehler) beschränkt.

Dabei wurde folgende Form gewählt:	normale Roman-Schrift	Ausgaben des Rechners
	fette SansSerif-Schrift	**Erforderliche Eingaben**
	kursive Roman-Schrift	*Beschreibungen und Anmerkungen*
	`Draft-Schrift`	`Auszüge aus Quellfiles und deren Änderungen`

Bauen der Tools und der Umgebung :

a) Übersetzung unter SOLARIS 2.3 (durchgeführt auf samson.hrz.tu-chemnitz.de)

```
samson% make IDENT=sun bindir
cd binsrc; make IDENT=sun VERSION=0.0 install
if [ "/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29" = "" ] ; then \
        echo "FATAL:  SPEC environment variable not set" ; \
        echo "Exiting tool make!!!" ; \
       exit 127 ; \
else \
        echo "Setting up /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29 directories now." ; \
fi
Setting up /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29 directories now.
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bin ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bin; fi
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man; fi
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1 ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1; fi
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3 ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3; fi
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman; fi
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/lib ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/lib; fi
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/include ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/include; fi
for i in lib cmd catman ; \
do \
      if ["$i" !="."] ; \
      then \
           cd$i ; \
           echo$i ; \
           if [ -r M.sun ] ; \
           then \
                     make -f M.sun IDENT=sun VERSION=0.0 install ; \
             elif [ -r Makefile -o -r makefile ] ; \
           then \
                     make IDENT=sun VERSION=0.0 install ; \
          fi ; \
          cd .. ; \
     fi ; \
done
lib
cc -I../include -O  -c  max.c
cc -I../include -O  -c  mstrcpy.c
cc -I../include -O  -c  strcar.c
cc -I../include -O  -c  strcdr.c
cc -I../include -O  -c  min.c
cc -I../include -O  -c  efopen.c
cc -I../include -O  -c  fgetsmod.c
cc -I../include -O  -c  getopt.c
cc -I../include -O  -c  traverse.c
"traverse.c", line 29: cannot find include file: <sys/dir.h>
"traverse.c", line 46: incomplete struct/union/enum direct: entry
"traverse.c", line 50: incomplete struct/union/enum direct: sizeof()
"traverse.c", line 51: undefined struct/union member: d_ino
"traverse.c", line 59: undefined symbol: DIRSIZ
"traverse.c", line 59: integral constant expression expected
"traverse.c", line 60: undefined struct/union member: d_name
"traverse.c", line 85: undefined symbol: DIRSIZ
"traverse.c", line 85: integral constant expression expected
cc: acomp failed for traverse.c
*** Error code 2
make: Fatal error: Command failed for target 'traverse.o'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/lib
*** Error code 1
make: Fatal error: Command failed for target 'all'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc
*** Error code 1
make: Fatal error: Command failed for target 'bindir'
samson%
```

Ändern von $SPECHOME/binsrc/lib/traverse.c

Zeile 29:

```
 #include <sys/dir.h> ---> #include <sys/fs/ufs_fsdir.h>
```

und vor 28 (alt) eine Zeile einfügen:

```
 #include <sys/param.h>


 27  #include <stdio.h>
 28  #include <sys/types.h>
 29  #include <sys/dir.h>
--->
 27  #include <stdio.h>
 28  #include <sys/param.h>
 29  #include <sys/types.h>
 30  #include <sys/fs/ufs_fsdir.h>
```

Neuer Versuch:

```
samson% make IDENT=sun bindir
...
cc -I../include -O  -c  traverse.c
"traverse.c", line 73: undefined symbol: DIR
"traverse.c", line 73: undefined symbol: dirp
"traverse.c", line 74: syntax error before or at: struct
"traverse.c", line 77: undefined symbol: filetype
"traverse.c", line 84: undefined symbol: entry
"traverse.c", line 87: left operand of "->" must be pointer to struct/union
"traverse.c", line 97: cannot recover from previous errors
cc: acomp failed for traverse.c
*** Error code 2
make: Fatal error: Command failed for target 'traverse.o'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/lib
*** Error code 1
make: Fatal error: Command failed for target 'all'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc
*** Error code 1
make: Fatal error: Command failed for target 'bindir'
samson%
```

---> Kopieren von fehlenden Defininionen (Zeile 39 sowie Zeilen 41-54 (alle Zeilennummern bzgl. des alten Files) aus dem else- in den if-Zweig ab Zeile 39 (neu))

```
  ...
    37  #else
    38
    39  #define DIR     FILE
    40  #define MAXNAME (DIRSIZ+2)
    41  #define opendir(path) fopen (path, "r")
    42  #define closedir(dirp) fclose (dirp)
    43  struct direct *
```

```
    44  readdir (dirp)
    45  DIR     *dirp;
    46          {
    47          static  struct  direct  entry;
    48          if (dirp == NULL) return (NULL);
    49          for (;;)
    50                  {
    51                  if (fread (&entry, sizeof (struct direct), 1, dirp) == 0) return (NULL);
    52                  if (entry.d_ino) return (&entry);
    53                  }
    54          }
    55  char    *strncpy ();
    56  char *
    57  namedir (entry)
    58  struct  direct  *entry;
    59          {
    60          static  char    name[MAXNAME];
    61          return (strncpy (name, entry->d_name, DIRSIZ));
    62          }
    63
    64  #endif
```

---> neu:

```
    32  #ifdef MAXNAMLEN
    33
    34  #define namedir(entry) (entry->d_name)
    35  #define MAXNAME 256
    36  #define DIR     FILE
    37  #define opendir(path) fopen (path, "r")
    38  #define closedir(dirp) fclose (dirp)
    39  struct direct *
    40  readdir (dirp)
    41  DIR     *dirp;
    42          {
    43          static  struct  direct  entry;
    44          if (dirp == NULL) return (NULL);
    45          for (;;)
    46                  {
    47                  if (fread (&entry, sizeof (struct direct), 1, dirp) == 0) return (NULL);
    48                  if (entry.d_ino) return (&entry);
    49                  }
    50          }
    51
    52  #else
    53
    54  #define DIR     FILE
    55  #define MAXNAME (DIRSIZ+2)
    56  #define opendir(path) fopen (path, "r")
    57  #define closedir(dirp) fclose (dirp)
    58  struct direct *
    59  readdir (dirp)
    60  DIR     *dirp;
    61          {
    62          static  struct  direct  entry;
    63          if (dirp == NULL) return (NULL);
    64          for (;;)
    65                  {
    66                  if (fread (&entry, sizeof (struct direct), 1, dirp) == 0) return (NULL);
    67                  if (entry.d_ino) return (&entry);
    68                  }
    69          }
    70  char    *strncpy ();
    71  char *
    72  namedir (entry)
    73  struct  direct  *entry;
    74          {
    75          static  char    name[MAXNAME];
    76          return (strncpy (name, entry->d_name, DIRSIZ));
    77          }
    78
    79  #endif
```

Damit ist dieses Problem bei der Installation (unter SOLARIS2.3) gelöst:

```
samson% make IDENT=sun bindir

...
lib
cc -I../include -O  -c traverse.c
sed -e "s/extern//" -e "s/\/\*\///g" < version.h > version.c
cc -c -I../include -O version.c
rm -f version.c
cc -I../include -O  -c strstr.c
cc -I../include -O  -c toupper.c
ar cr tools.a max.o mstrcpy.o strcar.o strcdr.o min.o  efopen.o fgetsmod.o getopt.o traverse.o version.o strstr.o toupper.o
touch tools.a
for i in tools.a ; do cp $i /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/lib ; done
for i in max.3 min.3 strcar.3 strcdr.3 mstrcpy.3 efopen.3s fgetsmod.3s  getopt.3 traverse.3 strstr.3 toupper.3 ; do cp $i /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3 ; done
cmd
for i in benchrun bin crc16 eval_tools head mount rantext spiff specclear ; \
do \
     if ["$i" != "."] ; \
     then \
         cd $i ; \
         echo $i ; \
         if [ -r M.sun ] ; \
         then \
                 make -f M.sun "IDENT=sun VERSION=0.0" install ; \
           elif [ -r Makefile -o -r makefile ] ; \
         then \
                         make "IDENT=sun VERSION=0.0" install ; \
        fi ; \
        cd .. ; \
    fi ; \
done
benchrun
cp benchrun.sh benchrun
cp benchthru.sh benchthru
chmod 755 benchrun
chmod 755 benchthru
bin
cp specman.sh specman
chmod 755 specman
cp specmanlist.sh specmanlist
chmod 755 specmanlist
cp sum_times.sh sum_times
chmod 755 sum_times
```

```
cp run_spec.sh run_spec
chmod 755 run_spec
if [ "/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29" = "" ] ; then \
        echo "WARNING: SPEC environment variable not set, install of specman specmanlist sum_times run_spec not performed" ; \
else \
        cp specman specmanlist sum_times run_spec /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bin ; \
        for i in specman specmanlist sum_times run_spec ; \
                do cp $i.1 /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1 ; \
      done ; \
fi
crc16
cc  -O   -c  crc16.c
cc  -O  -o crc16 crc16.o
eval_tools
cc  -I../../include  -O  -o convert_hms convert_hms.c
cc  -I../../include  -O  -o find_best find_best.c
cc  -I../../include  -O  -o time_diff time_diff.c
cc  -I../../include  -O  -o specmark specmark.c -lm
cc  -I../../include  -O  -o convert_ten convert_ten.c -lm
head
cc  -O   -c  head.c
cc  -O  -o head head.o
mount
if [ "/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29" = "" ] ; then \
        echo "WARNING:  SPEC environment variable not set" ; \
        echo "doSPECmount doSPECumount not installed" ; \
else \
        for i in doSPECmount doSPECumount ; \
      do \
                cp $i /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bin/$i; \
                chmod +x /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bin/$i ; \
      done \
fi
rantext
cc  -O   -c  rantext.c
cc  -O  -o rantext rantext.o
cc  -O   -c  asciicount.c
cc  -O  -o asciicount asciicount.o
cp textcount.SH textcount
cp printable.SH printable
if [ "/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29" = "" ] ; then \
        echo "WARNING: SPEC environment variable not set, install of rantext not performed" ; \
else \
        chmod +x textcount printable ; \
        cp rantext asciicount textcount printable /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bin ; \
        for i in rantext asciicount textcount printable ; \
                do cp $i.1 /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1 ; \
      done ; \
fi
spiff
cc -O  -DM_TERMCAP    -c  spiff.c
cc -O  -DM_TERMCAP    -c  output.c
compiler may report 1 statement not reached warning message for compare.c
cc -O  -DM_TERMCAP    -c compare.c
"compare.c", line 40: warning: statement not reached
cc -O  -DM_TERMCAP    -c  float.c
cc -O  -DM_TERMCAP    -c  strings.c
cc -O  -DM_TERMCAP    -c  exact.c
cc -O  -DM_TERMCAP    -c  miller.c
compiler may report 4 statement not reached warning messages for parse.c
if test -f /vmunix; \
then cc -O  -DM_TERMCAP    -c parse.c ; \
else cc -O  -DM_TERMCAP    -Dindex=strchr -c parse.c ; \
fi
"parse.c", line 286: warning: statement not reached
"parse.c", line 366: warning: statement not reached
"parse.c", line 477: warning: statement not reached
"parse.c", line 751: warning: statement not reached
cc  -O  -DM_TERMCAP    -c  command.c
cc  -O  -DM_TERMCAP    -c  comment.c
cc  -O  -DM_TERMCAP    -c  tol.c
cc  -O  -DM_TERMCAP    -c  line.c
cc  -O  -DM_TERMCAP    -c  token.c
cc  -O  -DM_TERMCAP    -c  floatrep.c
cc  -O  -DM_TERMCAP    -c  misc.c
cc -c -O  -DM_TERMCAP    visual.c
cc -O  -DM_TERMCAP   -o spiff spiff.o output.o compare.o float.o strings.o exact.o miller.o parse.o command.o comment.o tol.o line.o token.o floatrep.o misc.o visual.o  -ltermcap
specclear
make "EXTRA_CFLAGS=-DCLEAR=clear" install
cc  -O -DCLEAR=clear  -c  specclear.c
"specclear.c", line 15: undefined symbol: clear
cc: acomp failed for specclear.c
*** Error code 2
make: Fatal error: Command failed for target 'specclear.o'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/cmd/specclear
*** Error code 1
make: Fatal error: Command failed for target 'install'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/cmd/specclear
*** Error code 1
make: Fatal error: Command failed for target 'install'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/cmd
*** Error code 1
make: Fatal error: Command failed for target 'all'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc
*** Error code 1
make: Fatal error: Command failed for target 'bindir'
samson%
```

Editieren von $SPECHOME/binsrc/cmd/specclear/specclear.c :

```
 15           system(CLEAR);
--->
 15           system("clear");
```

```
samson% make IDENT=sun bindir

…
specclear
make "EXTRA_CFLAGS=-DCLEAR=clear" install
cc  -O -DCLEAR=clear  -c  specclear.c
cc  -O   -O -DCLEAR=clear -o specclear specclear.o
catman
if [ "/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29" = "" ] ; then \
        echo "WARNING: SPEC environment variable not set, install of PEC/man/catman not performed" ; \
else \
        for manpage in `ls /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1` ; \
```

```
      do \
                cat /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/tmac/specmac /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1/$manpage | nroff -Tlp | col -b > /a/zuse-
f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman/$manpage; \
      done ; \
        for manpage in 'ls /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3' ; \
      do \
                cat /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/tmac/specmac /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3/$manpage | nroff -Tlp | col -b > /a/zuse-
f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman/$manpage; \
      done \
fi
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir; fi
touch bindir

samson%
```

Die Installation und Übersetzung der Tools ist damit beendet.

b) Installation unter SunOS 4.1.3. (durchgeführt auf herkules.hrz.tu-chemnitz.de)

```
herkules-27 % make IDENT=sun bindir

...
cc -O  -DM_TERMCAP  -o spiff spiff.o output.o compare.o float.o strings.o exact.o miller.o parse.o command.o comment.o tol.o line.o token.o floatrep.o misc.o visual.o  -ltermcap
specclear
make "EXTRA_CFLAGS=-DCLEAR=clear" install
cc  -O -DCLEAR=clear  -target sun4 -c  specclear.c
"specclear.c", line 15: clear undefined
*** Error code 1
make: Fatal error: Command failed for target 'specclear.o'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/cmd/specclear
*** Error code 1
make: Fatal error: Command failed for target 'install'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/cmd/specclear
*** Error code 1
make: Fatal error: Command failed for target 'install'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/cmd
*** Error code 1
make: Fatal error: Command failed for target 'all'
Current working directory /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc
*** Error code 1
make: Fatal error: Command failed for target 'bindir'
herkules-28 %
```

Editieren von $SPECHOME/binsrc/cmd/specclear/specclear.c

```
15                system(CLEAR);
--->
15                system("clear");
```

```
herkules-28 % make IDENT=sun bindir

...
spiff
specclear
make "EXTRA_CFLAGS=-DCLEAR=clear" install
cc  -O -DCLEAR=clear  -target sun4 -c  specclear.c
cc  -O   -O -DCLEAR=clear -o specclear specclear.o
catman
if [ "/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29" = "" ] ; then \
        echo "WARNING: SPEC environment variable not set, install of PEC/man/catman not performed" ; \
else \
        for manpage in 'ls /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1' ; \
      do \
                cat /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/tmac/specmac /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1/$manpage | nroff -Tlp | col -b > /a/zuse-
f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman/$manpage; \
      done ; \
        for manpage in 'ls /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3' ; \
      do \
                cat /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/tmac/specmac /a/zuse-f
/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3/$manpage | nroff -Tlp | col -b > /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman/$manpage; \
      done \
fi
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir; fi
touch bindir
herkules-29 %
```

Die Installation und Übersetzung der Tools ist damit beendet.

Die späteren Messungen zeigen ein besonderes Format bei den Ausgaben des UNIX-Kommandos time, unter SunOS 4.1.3 werden die 3 Werte real, user und sys auf einer Zeile ausgegeben, wobei die Zeiten auch noch vorm entsprechenden Kürzel stehen. Dies wird in den einzelnen Benchmark-Makefiles nicht berücksichtigt, die awk-Zerlegung liefert dort keinen Zeitwert, so daß auch keine Bildschirmausgabe der Benchmarklaufzeit erfolgt. Dies läßt sich für dieses System korrigieren, indem man in jedem Benchmarkverzeichnis $SPECHOME/benchspec/<BENCHMARK> jeweils die Zeile mit dem awk-Aufruf in Makefile beim Ziel "compare" folgendermaßen verändert:

```
        -@time='awk '/^real/ {print $$2}' ./time.out'           ;\
--->
        -@time='awk '/real/ {print $$1}' ./time.out'            ;\
```

Entfernt man noch eines der Dollar-Zeichen, so werden alle 3 Zeitwerte ausgegeben.

c) Installation unter IRIX 5.2 (durchgeführt auf silly.hrz.tu-chemnitz.de)

```
silly-33 % make IDENT=sgi bindir
       cd binsrc; make IDENT=sgi VERSION=0.0 install
...
...
...
       if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir; fi
      touch bindir
silly-34 %
```

Die gesamte Installation war ohne Änderungen erfolgreich, es sind keine Fehler aufgetreten. Für Silicon Graphics sind in dieser Version keine herstellerspezifischen Überlagerungsfiles vorhanden, man kann also von einer "Standard-Installation" sprechen.

d) Installation unter HP-UX 9.01 (durchgeführt auf tantalus.hrz.tu-chemnitz.de)

```
tantalus-33 % make IDENT=hp validate
        cd binsrc; make IDENT=hp VERSION=0.0 install
...
...
spiff
        cc -O -DATT -DM_TERMINFO  -c spiff.c
        cc -O -DATT -DM_TERMINFO  -c output.c
compiler may report 1 statement not reached warning message for compare.c
        cc -O -DATT -DM_TERMINFO  -c compare.c
        cc -O -DATT -DM_TERMINFO  -c float.c
        cc -O -DATT -DM_TERMINFO  -c strings.c
        cc -O -DATT -DM_TERMINFO  -c exact.c
        cc -O -DATT -DM_TERMINFO  -c miller.c
compiler may report 4 statement not reached warning messages for parse.c
        cc -O -DATT -DM_TERMINFO  -c parse.c
        cc -O -DATT -DM_TERMINFO  -c command.c
        cc -O -DATT -DM_TERMINFO  -c comment.c
        cc -O -DATT -DM_TERMINFO  -c tol.c
        cc -O -DATT -DM_TERMINFO  -c line.c
        cc -O -DATT -DM_TERMINFO  -c token.c
        cc -O -DATT -DM_TERMINFO  -c floatrep.c
        cc -O -DATT -DM_TERMINFO  -c misc.c
        cc -c -O -DATT -DM_TERMINFO  visual.c
        cc -O -DATT -DM_TERMINFO  -o spiff spiff.o output.o compare.o float.o strings.o exact.o miller.o parse.o command.o comment.o tol.o line.o token.o floatrep.o misc.o visual.o  -lcurses
      mv spiff ../../bin
      cp spiff.1 ../../man/man1
cp: cannot create ../../man/man1: No such file or directory
*** Error code 1

Stop.
*** Error code 1

Stop.
*** Error code 1

Stop.
*** Error code 1

Stop.
tantalus-34 %
```

Hier liegt ein Fehler im verwendeten herstellerspezifischen Überlagerungsfile $SPECHOME/binsrc/cmd/spiff/M.hp vor. Die Variable INSDIR wird falsch gesetzt, so daß das Manual spiff.1 nicht ins SPEC-Manual-Verzeichnis $SPECHOME/man/man1 kopiert werden kann, da in der Verzeichnisstruktur nicht weit genug aufgestiegen wird. Das ausführbare Vergleichsprogramm spiff wird außerdem zwar kopiert, jedoch nicht ins Binary-Verzeichnis $SPECHOME/bin, sondern nach $SPECHOME/binsrc/bin. Dort wird es dann bei den eigentlichen Benchmarkläufen nicht gefunden, diese werden dann ergebnislos abgebrochen, da kein Vergleich mit SPEC-Referenzergebnissen erfolgen kann..
Abhilfe durch editieren von $SPECHOME/binsrc/cmd/spiff/M.hp :

```
      10  INSDIR=../..
--->
      10  INSDIR=../../..


tantalus-35 % make IDENT=hp bindir

...
spiff
       cc -O -DATT -DM_TERMINFO  -o spiff spiff.o output.o compare.o float.o strings.o exact.o miller.o parse.o command.o comment.o tol.o line.o token.o floatrep.o misc.o visual.o  -lcurses
      mv spiff ../../../bin
       cp spiff.1 ../../../man/man1
specclear
       cc  -O  -c specclear.c
       cc  -O   -O  -o specclear specclear.o
catman
       if [ "/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29" = "" ] ; then \
               echo 'WARNING: SPEC enviroment variable not set, install of PEC/man/catman not performed" ; \
      else \
               for manpage in `ls /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1` ; \
          do \
                       cat /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/tmac/specmac /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1/$manpage | nroff -Tlp | col -b > /a/zuse-
f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman/$manpage; \
          done ;\
               for manpage in `ls /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3` ; \
          do \
                       cat /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/tmac/specmac /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3/$manpage | nroff -Tlp | col -b > /a/zuse-
f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman/$manpage; \
          done \
      fi
       if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir; fi
       touch bindir
tantalus-36 %
```

Die Installation und Übersetzung der Tools ist damit beendet. Es wurde eine ganze Reihe herstellerspezifische M.hp-Files verwendet, die offensichtlich schon eventuelle Komplikationen bei der Installation beseitigen, so wie das bei der offiziellen Release eigentlich für alle Systeme der Fall sein sollte.

e) Installation unter Digital UNIX 3.2 (ehem. OSF/1) (durchgeführt auf decency.hrz.tu-chemnitz.de)

```
decency.hrz.tu-chemnitz.de-33 % make IDENT=dec bindir
cd binsrc; make IDENT=dec VERSION=0.0 install
...
...
cc -I../include -O  -c traverse.c
/usr/lib/cmplrs/cc/cfe: Warning: traverse.c, line 83: Incompatible pointer type assignment
                        while (entry = readdir (dirp))
                       ________^
/usr/lib/cmplrs/cc/cfe: Error: traverse.c, line 86: 'd_name' undefined, reoccurrences will not be reported
                            (void) strcpy (name, (entry->d_name));
                           ____________________^
/usr/lib/cmplrs/cc/cfe: Error: traverse.c, line 86: member of structure or union required
                            (void) strcpy (name, (entry->d_name));
                           ____________________^
*** Exit 1
Stop.
*** Exit 1
Stop.
*** Exit 1
Stop.
decency.hrz.tu-chemnitz.de-34 %
```

Editieren von $SPECHOME/binsrc/lib/traverse.c:

```
    29  #include <sys/dir.h>
--->
    29  #include <dirent.h>
```

(Es gibt in diesem System kein entsprechendes Headerfile <sys/dir.h>. Statt dessen gibt es dieses <dirent.h>, also wird dieses einbezogen.)
Außerdem muß auf Zeile 73 die Vereinbarung verändert werden. entry wird damit entsprechend der dirent-Definition in dirent.h vereinbart.

```
    73          struct  direct  *entry;
--->
    73          struct  dirent  *entry;
```

```
decency.hrz.tu-chemnitz.de-41 % make IDENT=dec bindir
...
lib
cc -I../include -O  -c traverse.c
sed -e "s/extern//" -e "s/\/\*\//g" < version.h > version.c
cc -c -I../include -O  version.c
...
specclear
cc  -O  -c specclear.c
cc  -O   -O  -o specclear specclear.o
sh: syntax error at line 1: 'end of file' unexpected
*** Exit 2
Stop.
*** Exit 1
Stop.
*** Exit 1
Stop.
*** Exit 1
Stop.
decency.hrz.tu-chemnitz.de-43 %
```

Editieren von $SPECHOME/binsrc/cmd/specclear/Makefile. Offensichtlich liegt hier ein Problem bei der Abarbeitung dieses Makefiles im Make-Mechanismus vor. Die 3 folgenden Kommentarzeilen werden fälschlicherweise als unerwartetes Fileende interpretiert. Durch Löschen der 3 sowieso auskommentierten Anweisungszeilen läßt sich das Problem lösen, so daß das Makefile vollständig durchgearbeitet wird.

```
---> Löschen von :
     72  #               for i in ${TOOLNAME} ; \
    73  #                        do cp $$i.1 ${SPEC}/man/man1 ; \
    74  #               done ; \
```

```
decency.hrz.tu-chemnitz.de-47 % make IDENT=dec bindir
cd binsrc; make IDENT=dec VERSION=0.0 install
...
spiff
specclear
catman
if [ "/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29" = "" ] ; then  echo "WARNING: SPEC environment variable not set, install of PEC/man/catman not performed" ;  else  for manpage in `ls
/a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1` ;  do  cat /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/tmac/specmac /a/zuse-
f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man1/$manpage | nroff -Tlp | col -b > /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman/$manpage;  done ;  for manpage in `ls /a/zuse-
f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3` ;  do  cat /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/binsrc/tmac/specmac /a/zuse-
f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/man3/$manpage | nroff -Tlp | col -b > /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/man/catman/$manpage;  done  fi
if [ ! -d /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir ] ; then mkdir /a/zuse-f/home/urz/fs22/c/cmun/Projekt/CFP95.29/bindir; fi
touch bindir
decency.hrz.tu-chemnitz.de-48 %
```

Die Installation und Übersetzung der SPEC-Tools für diese Plattform ist damit beendet.

Damit ist die Installation der gesamten Quellen und die Compilierung der bisher verfügbaren Verarbeitungswerkzeuge für alle Testsysteme erfolgreich abgeschlossen. Hierbei wurde nur Wert auf fehlerfreie Übersetzbarkeit gelegt. Auch während der eigentlichen Benchmarkläufe wurden keine weiteren Fehler in der (allerdings nur teilweise benutzten) Tool-Umgebung festgestellt.

Für 3 der Einzelbenchmarks sind spezielle Datenbasen (.MODEL-Files) erforderlich, die aus Speicherplatzgründen im Paket nicht direkt enthalten sind, sondern erst generiert werden müssen.. Dieser Schritt wird im folgenden Teil II dieser Anlage am Beispiel SOLARIS2.3 (durchgeführt auf samson.hrz.tu-chemnitz.de) gezeigt, für alle anderen Testsysteme läuft dieser Schritt analog ab. Diese .MODEL-Files enthalten i.A. die Initialisierungsdaten für große in den entsprechenden Einzelbenchmarks verwendete Felder.

Die 10 Benchmarks selbst werden vor jedem Lauf mit dem auf einer Maschine verfügbaren Compiler und den speziellen in einem Überlagerungsfile zum Makefile anzugebenden Optimierungsflags übersetzt und anschließend gestartet. Diese Vorgänge wurden bereits im Abschnitt 2.3. beschrieben.

Anlage A Teil II - Generierung erforderlicher Benchmark-Datenbasen

Aus Speicherplatzgründen wurden für 3 Benchmarks die erforderlichen Initialisierungsfiles nicht direkt mitgeliefert, sondern diese müssen erst generiert werden. Dies wird hier am Beispiel SOLARIS 2.3 (auf SAMSON) gezeigt. Es wird davon ausgegangen, daß man sich im Benchmark-Wurzelverzeichnis befindet(, nach der Installation ist dies das Verzeichnis CFP95.29). Die Umgebungsvariablen müssen mit source cshrc gesetzt worden sein.

Benchmark 101.tomcatv:

```
samson% cd benchspec
samson% cd 101.tomcatv
samson% cd src.model
samson% make ref
f77  -o genmodel genmodel.f
genmodel.f:
 MAIN:
genmodel < ../input.ref/tomcatv.in
mv ./TOMCATV.MODEL ../input.ref
samson% cd ../../..
samson%
```

Das erzeugte TOMCATV.MODEL-File befindet sich im input.ref-Verzeichnis im Benchmarkverzeichnis.

```
samson% ls -l ./benchspec/101.tomcatv/input.ref
total 21106
-rw-r--r--   1 cmun     user     10789929 Jul 24 13:02 TOMCATV.MODEL
-rwxr-xr-x   1 cmun     user         96 Jan 31 13:15 tomcatv.in
samson%
```

Benchmark 103.su2cor:

```
samson% cd benchspec
samson% cd 103.su2cor
samson% cd src.model
samson% make ref
f77  -o genmodel genmodel.f
genmodel.f:
 MAIN su2cor:
     init:
      trmgv:
     entry   trmget:
     entry   trmset:
       geom:

      clear2:
      clear4:
genmodel < ../input.ref/su2cor.in
 SPEC benchmark 103.su2cor

 Generation of initial model
mv ./SU2COR.MODEL ../input.ref
samson% cd ../../..
```

Das erzeugte SU2COR.MODEL-File befindet sich im input.ref-Verzeichnis im Benchmarkverzeichnis.

```
samson% ls -l ./benchspec/103.su2cor/input.ref
total 34866
-rw-r--r--   1 cmun     user     17825792 Jul 24 13:07 SU2COR.MODEL
-rwxr-xr--   1 cmun     user         39 Feb 21 21:44 su2cor.in
samson%
```

Benchmark 104.hydro2d :

```
samson% cd benchspec
samson% cd 104.hydro2d
samson% cd src.model
samson% make ref
f77  -c inpdta.f
inpdta.f:
      inpdta:
f77  -c genmodel.f
genmodel.f:
 MAIN hyd2dm:
      inival:
       inimod:
      prepar:
/opt/F77/bin/f77 -o genmodel inpdta.o genmodel.o
sh: /opt/F77/bin/f77: not found
*** Error code 1
make: Fatal error: Command failed for target 'compile'
samson%
```

---> Editieren der Compilerspezifikationszeile im Makefile $SPECHOME/benchspecd/104.hydro2d/src.model/Makefile :

```
      F77 =/opt/F77/bin/f77
--->
      F77 = f77
```

```
samson% make ref
f77 -o genmodel inpdta.o genmodel.o
genmodel < ../input.ref/hydro2d.in
  ISTEP    =        400
  ISTOP    =        500
  MPROW    =         300
  NPROW    =         1
  NEGFL    =        0
  NEGCN    =         1000
  CFL      =          0.75000000000000
  VCT      =          0.50000000000000
  MQFLG    =        0
  NQFLG    =        0
  ZL       =      0
  ZU       =      0
  RL       =      C
  RU       =      0
mv ./HYDRO2D.MODEL ../input.ref
samson% cd ../../..
```

Das erzeugte HYDRO2D.MODEL-File befindet sich im input.ref-Verzeichnis im Benchmarkverzeichnis.

```
samson% ls -l  ./benchspec/104.hydro2d/input.ref
total 2312
-rw-r--r--   1 cmun     user     1350720 Oct 11 09:20 HYDRO2D.MODEL
-rwxr-xr-x   1 cmun     user        514 Jun 22 10:30 hydro2d.in
samson%
```

Damit ist auch die Erzeugung der notwendigen Datenbasen für 3 Einzelbenchmarks abgeschlossen.