HANSER

Lutz Fröhlich

Oracle 11g Performance Forecast

Aktuelle und zukünftige Performance-Probleme erkennen und vermeiden

ISBN-10: 3-446-41494-0
ISBN-13: 978-3-446-41494-5

Leseprobe

2

2 Ansatz und Vorgehen

Es gibt eine Reihe von Büchern und Veröffentlichungen, die sich mit dem Thema Kapazitätsplanung und -Management für Computersysteme beschäftigen. Viele bleiben jedoch im theoretischen Ansatz stecken oder vergnügen sich mit ausschweifenden mathematischen Theorien und Herleitungen.

Mit dem vorliegenden Buch wollen wir praktische Mittel und Wege aufzeigen, um Capacity Management und insbesondere Performance Monitoring und Forecasting in das tägliche Support-Geschäft zu integrieren. Das bedeutet keineswegs, dass mathematische Methoden unberücksichtigt bleiben. Im Gegenteil: sie sind ein hervorragendes Mittel, um innerhalb kurzer Zeit und mit wenig Kosten zuverlässige Vorhersagen zu treffen. Aus diesem Grund erläutern wir mathematische Methoden ausführlich und stellen sie anhand praktischer Beispiele dar.

Weil es immer auch um die Anwendbarkeit der Theorie geht, beginnen wir mit der Frage, wie Sie ein Capacity Management für Oracle Datenbanken im täglichen Betrieb wirksam einsetzen. Wenn Sie im Support-Umfeld tätig sind, wissen Sie, wie dynamisch man auf Probleme reagieren muss und wie kurzfristig Entscheidungen umgesetzt werden müssen, ohne alle Auswirkungen berücksichtigen zu können. So wird eine Datenbank aus den unterschiedlichsten Gründen mit einer Vielzahl von Änderungsanträgen konfrontiert. Treten Sie einen Schritt zurück, und versetzen Sie sich in die Lage der Datenbank, die natürlich immer bemüht ist, jede Anforderung mit dem geringstmöglichen Aufwand an Ressourcen und in kürzester Zeit abzuarbeiten.

2.1 Einflüsse auf die Performance

Eine Datenbank unterliegt während ihres Lebenszyklus ständigen Veränderungen. Diese Änderungen sind zu einem Großteil geplant, wenige sind ungewollt. Trotzdem bewirken sie über kurz oder lang, dass sich die Performance verschlechtert und möglicherweise kritische Ausmaße annimmt. Aus der Perspektive der Datenbank sind das einfach eine Vielzahl von Ereignissen und Änderungen:

- Datenbank-Upgrades und -Patches
- Änderung der Applikationen
- Änderungen im SQL- und PL/SQL-Code
- Veränderte Datenmenge
- Wachstum der Datenbank
- Veränderte Daten-Histogramme
- Anpassung der Schema-Objekte wie Indexe oder partitionierte Tabellen
- Geänderte Datenbank-Parameter
- Entstehung von Hot Spots für Tabellen und Datenbank-Dateien
- Fragmentierung von Tabellen und Indexen
- Bugs
- Veränderte Hardware-Ressourcen
- Upgrades und Patches des Betriebssystems oder Storage-Subsystems
- Änderung der Belastung von geteilten Ressourcen
- Veränderte Anzahl von gleichzeitigen Benutzern.

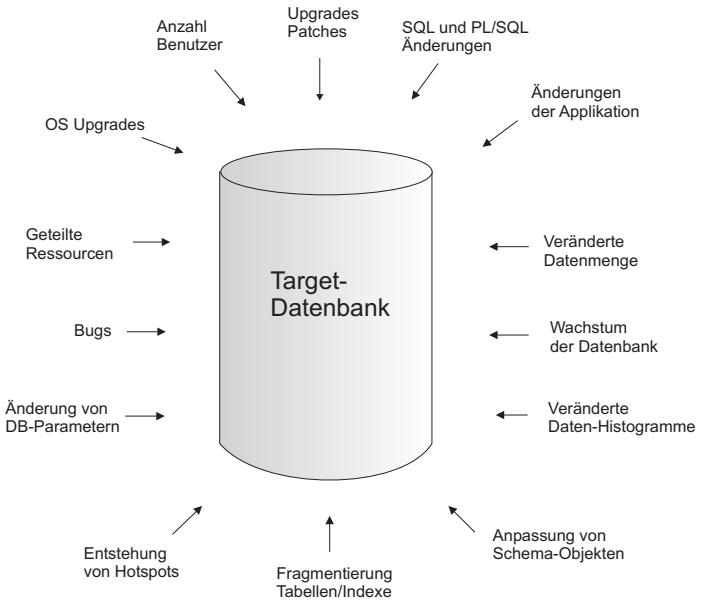

Abbildung 2.1 Einflüsse auf die Datenbank-Performance

Da Oracle interne Features wie den SQL-Optimizer ständig weiterentwickelt, führen Upgrades zu Veränderungen der Ausführungspläne. Möglicherweise wird nach einem Upgrade anstelle eines Index Scan ein Full Table Scan ausgeführt. Beim Upgrade auf die Version 10g konnte ein verändertes Verhalten im Ressourcen-Verbrauch bei parallelen SQL-Abfragen festgestellt werden. Plötzlich stieg die CPU-Auslastung bei Nested Loop-Plänen signifikant an.

Änderungen in der Applikation gehen nicht nur einher mit geänderten oder neuen SQL-Anweisungen. Auch veränderte Terminpläne nehmen Einfluss auf die Performance. Wenn plötzlich Batch Jobs oder Backups in den Tag hineinlaufen, nehmen sie Ressourcen weg, welche die Benutzer für ihre Online-Abfragen benötigen.

Eine temporäre Änderung in der Menge der zu verarbeitenden Daten und das Wachstum der Datenbank tragen einen großen Anteil an einer Verschlechterung der Performance. Sicherlich kennen Sie aus der Praxis die SQL-Anweisungen, die mit kleinen bis mittelgroßen Tabellen noch akzeptabel laufen, ab einer gewissen Tabellengröße jedoch Probleme verursachen. Auch eine Veränderung in den Histogrammen der Daten können den Optimizer veranlassen, Ausführungspläne zu verändern. Aber auch Verschiebungen in der Index-Struktur, Veränderungen des Layouts von partitionierten Tabellen oder Materialized Views in Zusammenhang mit dem Summary Management können zu erhöhtem Ressourcen-Verbrauch und verminderter Performance führen, ohne dass Änderungen im SQL-Quelltext selbst erfolgen.

Die Oracle-Datenbank ist bekannt für eine Vielzahl von änderbaren Parametern. So kann es als Folge einer Parameteränderung zu einer Verbesserung bei Online-Abfragen kommen, und gleichzeitig laufen die Batch Jobs aus dem Zeitplan hinaus. Außerdem gibt es die Kategorie von Einflüssen, die einfach dadurch entstehen, dass die Datenbank ohne Wartung über einen gewissen Zeitraum läuft. Dazu gehören die Fragmentierung von Tabellen und Indexen sowie das Entstehen von Hot Spots. Bugs sind besonders kritisch, da sie häufig nur in bestimmten Situationen auftreten und schwer zu identifizieren sind.

Aber auch Upgrades oder Parameter-Änderungen im Betriebssystem oder Storage-Bereich können zu erheblichen Veränderungen in der Ressourcen-Bereitstellung führen. Unterschätzt wird häufig der Sachverhalt, dass sich Datenbanken Ressourcen teilen, ohne dass ein Ressource Manager eingesetzt wird. Das gilt insbesondere für Storage-Systeme oder wenn sich Datenbanken einen Server teilen.

Ähnlich wie das Wachstum der Datenbank ist eine steigende Anzahl von gleichzeitigen Benutzern Hauptursache für einen plötzlichen Performance-Einbruch. Ich erinnere mich an einen Fall, als die Benutzeranzahl verdoppelte wurde und der Einfluss auf die Performance der Datenbank als gering eingeschätzt wurde, da die Benutzer-Sitzungen einen großen Anteil an Inaktivität auswiesen. Man hatte dabei vergessen, dass jeder Benutzer einen Serverprozess erzeugt, was letztendlich das Betriebssystem zum Paging veranlasste.

Sicher haben nicht alle Ereignisse einen negativen Einfluss auf die Performance oder führen zu einer signifikanten Verschlechterung. Die Praxis beweist jedoch, dass je länger eine Datenbank diesen Einflüssen unkontrolliert unterliegt, desto größer ist die Wahrscheinlichkeit, dass die Performance in den kritischen Bereich übergeht.

Es stellt sich die Frage: Ist es bei dieser Vielzahl von Einflüssen und der Komplexität des Datenbank-Betriebssystems überhaupt möglich, zuverlässige Vorhersagen zur Performance zu erstellen? Diese Frage wird in den folgenden Abschnitten untersucht.

2.2 Trennung von Change und Capacity Management

Eine saubere Trennung von Change und Capacity Management ist die erste und wichtigste Maßnahme für zuverlässige Performance-Vorhersagen. Ich erhielt kürzlich eine Anfrage mit der Aufgabe, eine Performance-Vorhersage zu erstellen. Der Kunde hatte ein Datenbank-Upgrade geplant und wollte wissen, wie die Performance der Applikation unter der neuen Version sein würde. Diesen Auftrag musste ich ablehnen, da es sich hierbei eindeutig um eine Aufgabe für das Change Management und nicht für das Capacity Management handelte. Halten Sie sich stets vor Augen, dass Vorhersagen immer auf einem Modell basieren. Modelle sind allerdings keine Eins-zu-eins-Abbildungen der Natur, sondern stellen eine Vereinfachung dar, in der die wichtigsten Einflussfaktoren berücksichtigt werden. Die internen Strukturen und Prozesse des Datenbank-Betriebssystems sind viel zu komplex, um in vertretbarer Zeit ein zuverlässiges Modell zu erstellen, das die Veränderungen eines Upgrades widerspiegelt.

Generell lässt sich sagen, dass der Einfluss durch Änderungen, die dem Change Management entstammen, effektiver, kostengünstiger und schneller durch reale Tests herausgefunden werden kann. Viele Firmen praktizieren dieses Vorgehen bereits, indem sie so genannte *UAT-Datenbanken* betreiben und vor der Implementierung in der Produktion die Auswirkungen auf die Applikation testen. Die Abkürzung *UAT* steht für *User Acceptance Testing*. Auch Oracle hat Bedeutung und Notwendigkeit eines solchen Vorgehens erkannt und in der Version 11g mit dem neuen Feature *Real Application Testing* darauf reagiert. Wir wollen uns an dieser Stelle jedoch nicht zu weit vom Thema entfernen.

Der Einfluss von durch das Change Management verursachten Änderungen ist jedoch real. Es stellt sich damit die Frage nach der Gültigkeit bereits gebildeter Prognosen, wenn Änderungen in der Datenbank im Rahmen des Change Managements vorgenommen werden. Generell lässt sich sagen, dass Prognosen ihre Gültigkeit behalten, solange die Änderungen keinen Einfluss auf den Workload haben. Die Überwachung des Workloads ist also ein wichtiger Bestandteil der fortlaufenden Validierung von Forecasts.

Jede Vorhersage basiert auf einer Baseline, die einen bestimmten Workload der Datenbank charakterisiert. Eine Veränderung der Baseline macht eine Neuberechnung der Vorhersage erforderlich. Das stellt insofern kein Problem dar, da die Neuberechnung mit mathematischen Methoden nicht aufwendig ist und in kurzer Zeit erfolgen kann. Allerdings müssen für die neue Prognose aktuelle Workload-Statistiken herangezogen werden. Viele Datenbanken lassen sich nach dem Prinzip „*Und täglich grüßt das Murmeltier*“ betrachten. Die Workload-Charakteristik wiederholt sich alle 24 Stunden, und damit können innerhalb von 24 Stunden die Statistiken für eine neue Baseline gesammelt werden. Für Datenbanken, die am Wochenende oder Monatsende spezifische Workloads ausweisen, gibt es Methoden, die Statistiken anzupassen, ohne auf Ergebnisse eines Langzeit-Sammelprozesses warten zu müssen.

Durch eine klare Trennung von Change Management und Capacity Management verringert sich die in Abbildung 2.2 dargestellte Komplexität von Einflussfaktoren entscheidend, und das Thema Performance Forecast wird insofern beherrschbar.

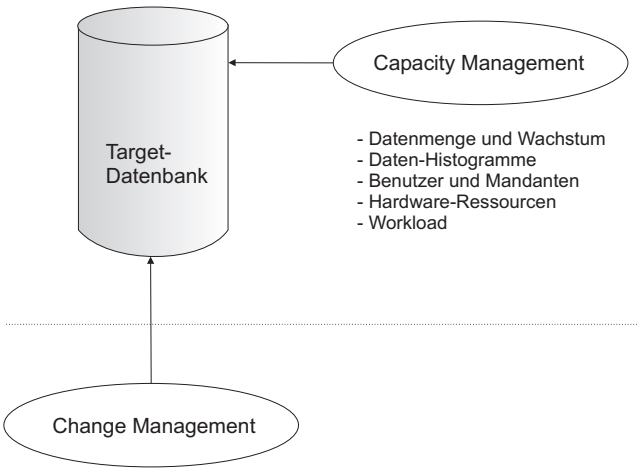

Abbildung 2.2
Trennung von Change Management und Capacity Management

Die Trennung von Change und Capacity Management bedeutet nicht, dass die Änderungen, die aus dem Change Management kommen, ignoriert werden. Durch den Einsatz von UAT-Datenbanken und Real Application Testing werden die Auswirkungen auf den Workload vorab bestimmt. Gleichzeitig garantiert das Workload Monitoring, dass Auswirkungen, die durch das Change Management entstehen, nicht unentdeckt bleiben.

2.3 Die Bedeutung von Workload-Statistiken

Workload-Statistiken sind eine notwendige Voraussetzung für das Erstellen von Performance Forecasts. Jede Prognose wird auf Basis einer Baseline gebildet. Eine Baseline ist ein charakteristischer Workload im betrachteten Zeitfenster. Die Statistiken bilden außerdem die Basis für die Schwellenwerte des Workload Monitoring. Forecast und Monitoring bedingen einander. Während ein Forecast die Werte für die erwartete Entwicklung des Workloads an das Monitoring liefert, sendet das Monitoring Warnungen und Alarmsignale über Abweichungen vom erwarteten Workload an den Forecast. Der Validierungsprozess untersucht die Abweichungen und stellt die Ursachen fest. Abweichungsursachen können zum Beispiel aus dem Change Management kommen oder durch ein überplanmäßiges Wachstum der Datenbank ausgelöst werden.

Hinweis
Auch wenn Sie für bestimmte Datenbanken keine Performance Forecasts erstellen wollen, können Sie das Performance Monitoring einsetzen, in dem Sie Vorgaben für den erwarteten Workload machen. Vom Monitoring erhalten Sie Alerts, wenn es zu Abweichungen kommt, und Sie können die Ursachen ermitteln und entsprechende Maßnahmen einleiten. Die frühzeitige Erkennung von Abweichungen unterstützt die Einhaltung von Performance-Vorgaben und Service Level Agreements. Gleichzeitig charakterisiert das Monitoring den Einfluss der Änderungen im Rahmen des Change Managements auf den Workload.

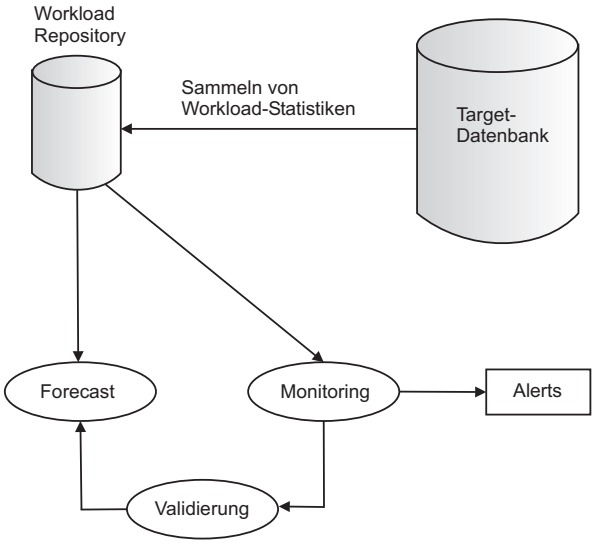

Abbildung 2.3 Prozess-Struktur von Forecast, Monitoring und Workload-Statistiken

Vor dem Sammeln von Workload-Statistiken muss festgelegt werden, welche Daten in welcher Häufigkeit gesammelt werden sollen. Behalten Sie dabei immer im Auge, dass mit dem Sammeln ein signifikanter Performance-Einfluss und zusätzlicher Ressourcen-Verbrauch auf der Target-Datenbank stattfinden kann. Andererseits benötigen Sie natürlich Statistikdaten in hinreichender Anzahl und guter bis sehr guter Qualität. Workload-Statistiken sind *rohe Daten*. Sie müssen aufbereitet werden, um daraus eine Baseline zu bilden und eine Basis für Vorhersagen zu liefern. Es handelt sich dabei um eine durchaus komplexe Aufgabe. Deshalb ist im Buch diesem Thema ein ganzes Kapitel gewidmet. Ausführliche Informationen finden Sie in Kapitel 3 „Workload-Statistiken“.

2.4 Ein Beispiel

An dieser Stelle soll ein Beispiel den Ansatz und das praktische Vorgehen illustrieren. Auch wenn Sie möglicherweise noch wenig über Forecast-Methoden wissen, können Sie leicht nachvollziehen, worum es bei dem gewählten Ansatz geht.

Es soll eine Vorhersage getroffen werden, ob auf einer Oracle-Datenbank, die eine Knowledge Base enthält, die Anzahl von Benutzern verdoppelt werden kann, ohne dass eine Erweiterung der zur Verfügung stehenden Hardware-Ressourcen erfolgt. Zur Zeit werden maximal 200 gleichzeitige Benutzer für die Online-Applikation zugelassen. Da die Datenbank über einen großen Buffer Cache verfügt, findet die Verarbeitung vorwiegend im Shared Memory statt, das heißt, die I/O-Aktivitäten können an dieser Stelle vernachlässigt werden. Von den Benutzern werden sehr häufig Volltext-Suchen durchgeführt und damit Full Table Scans in der Datenbank ausgeführt.

Eine Möglichkeit, die Frage zu beantworten, wäre, einen Benchmark durchzuführen und den Workload zu simulieren. Da aber kein Testsystem mit derselben Datenbankgröße und identischen Hardware-Ressourcen zur Verfügung steht und eine schnelle Entscheidung getroffen werden muss, soll ein Performance Forecast erstellt werden.

Der erste Schritt ist das Sammeln von Workload-Statistiken. Der Datenbank-Administrator entscheidet sich, Basisdaten zur Zeit der höchsten Systembelastung mit einer Häufigkeit von 10 Sekunden zu sammeln. Um den Workload zu charakterisieren, entschließt er sich, die Anzahl der *User Calls* heranzuziehen. Jede SQL-Anweisung löst eine gewisse Anzahl von User Calls auf der Datenbank aus. Es handelt sich dabei um eine kleine Transaktion, allerdings kann sie sehr universell betrachtet werden. Eine weitere benötigte Statistik ist natürlich die Anzahl der aktiven Benutzer auf der Datenbank.

Weiterhin wird natürlich die CPU-Auslastung als Statistik für den Ressourcen-Verbrauch benötigt. Der Administrator entschließt sich, die Werte für User- und System-CPU-Auslastung zu sammeln. Das folgende Skript ermittelt die Statistiken:

Listing 2.1 Beispiel für das Sammeln von Workload-Statistiken

```
#!/bin/ksh
# Sammeln von Workload-Statistiken
export ORACLE_HOME=/opt/oracle/product/11.1.0/db_1
export ORACLE_SID=hanser
export ORACLE_BASE=/opt/oracle
export PATH=$ORACLE_HOME/bin:$PATH
while true; do
sar 9 1 > /tmp/02_01.sar
USR_CPU=`cat /tmp/02_01.sar | tail -1 | awk '{ print $3 }'`
SYS_CPU=`cat /tmp/02_01.sar | tail -1 | awk '{ print $5 }'`
TOT_CPU=`cat /tmp/02_01.sar | tail -1 | awk '{ print $8 }'`
let TOT_CPU=${USR_CPU}+${SYS_CPU}
$ORACLE_HOME/bin/sqlplus -s / as sysdba <<EOF
INSERT INTO pf_simple VALUES (sysdate,(SELECT value FROM v\$sysstat WHERE
name='user calls'),(SELECT count(*) FROM v\$session WHERE status = 'ACTI-
VE' and username IS NOT NULL),$USR_CPU,$SYS_CPU,$TOT_CPU);
COMMIT;
EXIT;
EOF
done
```

Hinweis

Speichern Sie im produktiven Umfeld die Workload-Statistiken nicht in der Target-Datenbank. Dies stellt nicht nur eine Zusatzbelastung dar, sondern verfälscht möglicher weise das Ergebnis. Kapitel 3 beschäftigt sich ausführlich mit dem Thema „Sammeln von Workload-Statistiken“.

Die Tabellenkalkulation in Abbildung 2.4 zeigt die Ergebnisse des Sammelns von Workload-Daten. Es wurden teilweise mehr als 200 aktive Sessions aufgezeichnet. Dabei handelt es sich um administrative Benutzer, die keinen signifikanten Workload erzeugen.

	A	B	C	D	E	F
1	**Workload-Statistik Knowledge Base**					
2	SNAP_TIME	USER_CALLS	ACT_SESS	CPU_USER	CPU_SYS	CPU_TOTAL
3	12.02.2008_13:26:08	75424	191	26,73	9,70	36,43
4	12.02.2008_13:25:57	75343	232	35,57	12,32	47,89
5	12.02.2008_13:25:47	75311	231	37,71	12,04	49,75
6	12.02.2008_13:25:37	75265	241	35,03	11,38	46,41
7	12.02.2008_13:25:27	75218	233	35,08	13,04	48,12
8	12.02.2008_13:25:16	75186	233	36,02	11,48	47,50
9	12.02.2008_13:25:06	75137	242	35,26	11,46	46,72
10	12.02.2008_13:24:56	75087	233	35,72	12,24	47,96
11	12.02.2008_13:24:45	75051	231	40,25	12,93	53,18
12	12.02.2008_13:24:35	74942	239	35,57	11,17	46,74
13	12.02.2008_13:24:25	74895	228	38,07	14,00	52,07
14	12.02.2008_13:24:15	74851	227	36,83	11,68	48,51
15	12.02.2008_13:24:04	74796	241	35,28	12,15	47,43
16	12.02.2008_13:23:54	74749	227	35,70	12,13	47,83
17	12.02.2008_13:23:44	74707	228	36,44	11,89	48,33
18	12.02.2008_13:23:34	74665	238	38,51	12,38	50,89
19	12.02.2008_13:23:23	74541	227	34,15	13,05	47,20
20	12.02.2008_13:23:13	74506	231	36,58	12,84	49,42
21	12.02.2008_13:23:03	74450	232	43,03	11,77	54,80
22	12.02.2008_13:22:52	74405	236	38,25	15,34	53,59
23	12.02.2008_13:22:42	74335	231	38,26	14,09	52,35
24	12.02.2008_13:22:32	74224	181	26,08	8,92	35,00
25	12.02.2008_13:22:21	74179	192	26,86	9,22	36,08
26	12.02.2008_13:22:11	74145	188	28,81	8,98	37,79
27	12.02.2008_13:22:01	74090	182	26,42	8,64	35,06
28	12.02.2008_13:21:51	74049	191	29,26	12,02	41,28
29	12.02.2008_13:21:40	73961	178	35,16	10,60	45,76
30	12.02.2008_13:21:29	73875	181	26,40	9,67	36,07
31	12.02.2008_13:21:19	73756	189	24,86	10,13	34,99
32	12.02.2008_13:21:08	73637	191	28,99	9,66	38,65
33	12.02.2008_13:20:57	73522	132	18,65	7,62	26,27
34	12.02.2008_13:20:46	73396	134	18,84	7,15	25,99
35	12.02.2008_13:20:36	73339	141	22,62	6,02	28,64
36	12.02.2008_13:20:26	73229	129	18,16	6,19	24,35
37	12.02.2008_13:20:15	73182	128	23,74	6,46	30,20
38	12.02.2008_13:20:04	73142	133	21,23	6,15	27,38
39	12.02.2008_13:19:54	73090	127	19,33	6,71	26,04
40	12.02.2008_13:19:44	73044	131	18,81	7,41	26,22
41	12.02.2008_13:19:34	72998	181	28,82	9,93	38,75
42	12.02.2008_13:19:23	72906	179	27,43	10,62	38,05
43	12.02.2008_13:19:13	72764	178	23,59	8,51	32,10
44	12.02.2008_13:19:01	72670	129	19,58	7,85	27,43

Abbildung 2.4 Ergebnisse des Sammelns von Workload-Statistiken

Wie bereits erwähnt, handelt es sich bei den gesammelten Statistiken um Rohdaten. Diese müssen für das Erstellen eines Forecasts aufbereitet werden. Die aufbereiteten Daten werden als Baseline bezeichnet.

Es ist deutlich zu sehen, dass eine Abhängigkeit zwischen der Anzahl der aktiven Sessions und der CPU-Auslastung besteht. Andererseits sieht man auch Schwankungen in der CPU-Auslastung bei gleicher oder ähnlicher Anzahl von Sessions. Um einen besseren Eindruck von der Abhängigkeit der CPU-Auslastung zu erhalten, erstellen wir eine Punkt-Grafik, ein so genanntes *Scatterplot*. Jeder Punkt stellt einen Snapshot zu einem bestimmten Zeitpunkt dar.

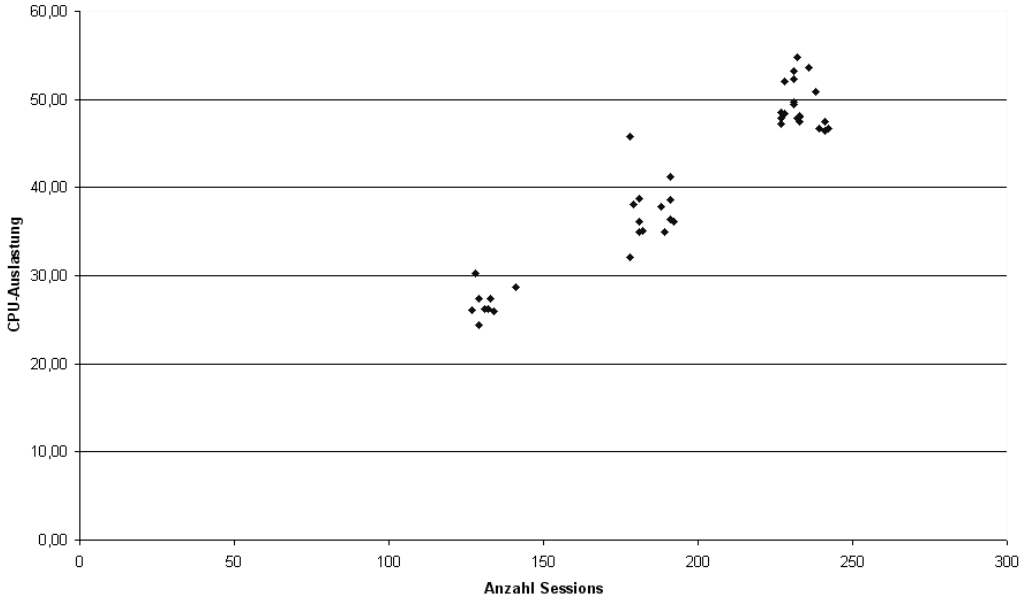

Abbildung 2.5 Scatterplot-Grafik der Workload-Statistiken

Ein kurzer Blick auf die Grafik genügt, um – mit gewissen Abweichungen – eine eindeutige Tendenz der Abhängigkeit der CPU-Auslastung von der Anzahl der Sessions zu konstatieren. Diesem Ergebnis gemäß beschließt der Datenbank-Administrator, ein einfaches mathematisches Modell für das Forecasting zu verwenden. Das Modell setzt voraus, dass ein Verhältnis zwischen der Anzahl von Sessions und der CPU-Auslastung besteht. Wird zusätzlich die mögliche Abweichung von der Ideallinie berücksichtigt, ist es möglich, eine durchaus zuverlässige Vorhersage zu treffen.

Die CPU-Auslastung (in Prozent) pro Session lässt sich Abbildung 2.4 dank einer zusätzlichen Tabellenspalte mühelos entnehmen. Der Wert berechnet sich einfach als Verhältnis von CPU-Auslastung und Anzahl der aktiven Sessions. Daraus ergibt sich das folgende Schema, in dem die übrigen Spalten entfernt wurden, da sie in diesem Zusammenhang ohne Bedeutung sind.

Die CPU-Auslastung pro Session schwankt zwischen 0,18 und 0,26 Prozent. Berechnen wir nun den statistischen Mittelwert für die CPU-Auslastung pro Session nach der folgenden Formel:

$$\overline{C} = \frac{1}{42}\left(\sum_{i=1}^{42} C_i\right) = \frac{1}{42} 8{,}71 = 0{,}20738$$

Der Mittelwert (gerundet) für die CPU-Auslastung aller gesammelten Statistiken beträgt 0,20738%. Es wird ein lineares Verhalten für die Berechnung angesetzt. Das Ziel besteht jedoch darin, dass zu jeder Zeit genügend CPU-Kapazitäten zur Verfügung stehen, ohne dass es zu Wartezeiten kommt. Im Extremfall muss damit gerechnet werden, dass die ma-

ximale Abweichung von der Ideallinie auftritt. Die maximale Abweichung nach oben beträgt 0,26 – 0,20738 = 0,05262 Prozent pro Session.

	A	B	C	D
1	SNAP_TIME	ACT_SESS	CPU_TOTAL	CPU_P_SES
2	12.02.2008_13:26:08	191	36,43	0,19
3	12.02.2008_13:25:57	232	47,89	0,21
4	12.02.2008_13:25:47	231	49,75	0,22
5	12.02.2008_13:25:37	241	46,41	0,19
6	12.02.2008_13:25:27	233	48,12	0,21
7	12.02.2008_13:25:16	233	47,50	0,20
8	12.02.2008_13:25:06	242	46,72	0,19
9	12.02.2008_13:24:56	233	47,96	0,21
10	12.02.2008_13:24:45	231	53,18	0,23
11	12.02.2008_13:24:35	239	46,74	0,20
12	12.02.2008_13:24:25	228	52,07	0,23
13	12.02.2008_13:24:15	227	48,51	0,21
14	12.02.2008_13:24:04	241	47,43	0,20
15	12.02.2008_13:23:54	227	47,83	0,21
16	12.02.2008_13:23:44	228	48,33	0,21
17	12.02.2008_13:23:34	238	50,89	0,21
18	12.02.2008_13:23:23	227	47,20	0,21
19	12.02.2008_13:23:13	231	49,42	0,21
20	12.02.2008_13:23:03	232	54,80	0,24
21	12.02.2008_13:22:52	236	53,59	0,23
22	12.02.2008_13:22:42	231	52,35	0,23

Abbildung 2.6 CPU-Auslastung pro Session

Mit diesen Zahlen ist der Datenbank-Administrator nun in der Lage, die Vorhersage zu erstellen. Die Frage war, ob es möglich ist, die Anzahl der Online-Sessions von 200 auf 400 zu verdoppeln. Zuerst wird der zu erwartende mittlere Wert für die CPU-Auslastung berechnet:

$$C = \overline{C} \times 400 = 0{,}20738 \times 400 = 82{,}952$$

Die maximale Abweichung berechnet sich wie folgt:

$$C_{\max} = 400 \times 0{,}05262 = 21{,}048$$

Die Addition der mittleren Auslastung und der maximalen Abweichung ergibt einen Wert von 104 Prozent. Dies bedeutet, dass die CPU zwar im Durchschnitt mit akzeptablen 82,952% belastet ist, im Extremfall aber mit 104% überlastet sein kann. Da jedoch Wartezeiten bei den Online-Anfragen unbedingt vermieden werden sollen, ist diese Zahl als zu hoch anzusehen. Der Administrator beschließt deshalb, die Kalkulation mit einem Ziel von 300 Sessions zu wiederholen, und erhält folgende Ergebnisse:

$$C = \overline{C} \times 300 = 0{,}20738 \times 300 = 62{,}214$$
$$C_{\max} = 300 \times 0{,}05262 = 15{,}786$$

Die Vorhersage der CPU-Auslastung für 300 Sessions liegt unter Berücksichtigung der maximalen Abweichung bei 78%. Das ist ein Wert, der ein gutes Systemverhalten garantiert und keine Wartezustände bei den Online-Abfrage erwarten lässt. Der Datenbank-

Administrator spricht die Empfehlung aus, die Anzahl der Online-Benutzer maximal auf 300 zu erhöhen. Um seine Prognose zu bestätigen, sammelt der Administrator Workload-Statistiken mit der freigegebenen Anzahl von 300 Sessions. In Abbildung 2.7 können Sie sehen, dass die CPU-Auslastung im vorhergesagten Bereich liegt. Auch die Schwankungen bewegen sich wie vorhergesagt, und die maximale Abweichung wurde nicht überschritten.

	A	B	C	D	E	F
1	**Workload-Statistiken Knowledge Base 300 Sessions**					
2	SNAP_TIME	USER_CALLS	ACT_SESS	CPU_USER	CPU_SYS	CPU_TOTAL
3	12.02.2008_18:47:36	150604	33	51,77	18,27	70,04
4	12.02.2008_18:47:25	150529	33	52,72	17,8	70,52
5	12.02.2008_18:47:15	150473	34	50,49	19,14	69,63
6	12.02.2008_18:47:04	150429	34	50,1	17,44	67,54
7	12.02.2008_18:46:54	150367	33	55,44	20,86	76,3
8	12.02.2008_18:46:43	150260	33	55,22	20,59	75,81
9	12.02.2008_18:46:32	150110	34	49,55	18,44	67,99
10	12.02.2008_18:46:20	150027	33	53,45	20,02	73,47
11	12.02.2008_18:46:08	149897	33	47,34	16,34	63,68
12	12.02.2008_18:45:57	149853	33	46,81	15,46	62,27
13	12.02.2008_18:45:45	149817	34	48,38	18,21	66,59
14	12.02.2008_18:45:32	149647	34	44,48	16,38	60,86
15	12.02.2008_18:45:17	149544	31	37,25	12,9	50,15
16	12.02.2008_18:45:04	149451	31	44,46	20,26	64,72

Abbildung 2.7 Workload-Statistiken mit 300 Online-Sessions

Schauen Sie sich auch hier wieder die Scatterplot-Grafik an. Sie vermittelt einen sehr guten optischen Eindruck über den Auslastungsbereich und die Abweichungen vom Durchschnitt.

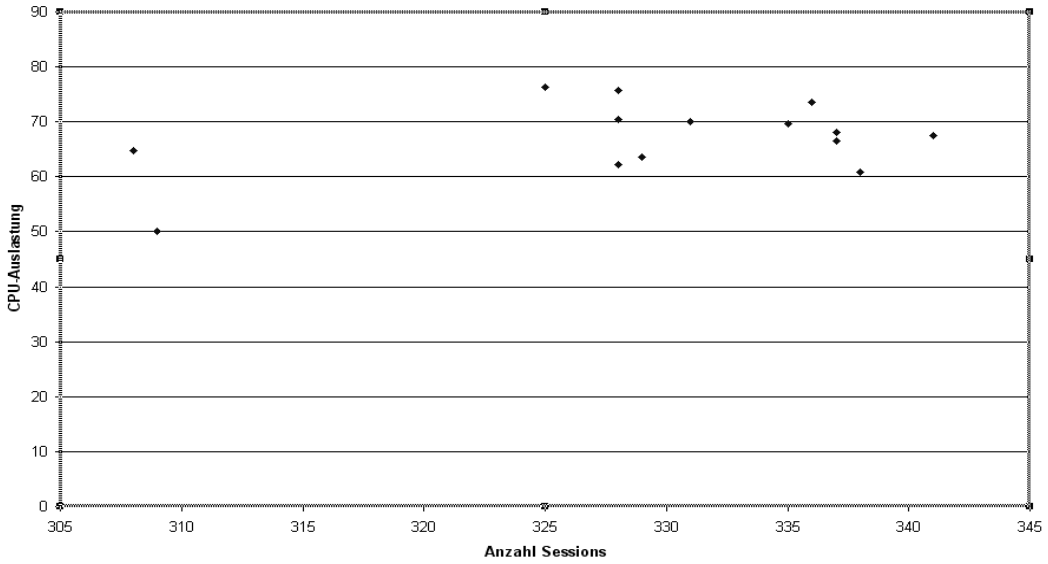

Abbildung 2.8 Scatterplot-Grafik für maximal 300 Online-Sessions

Mit der Erhöhung des Workloads auf maximal 300 Online-Sessions kommt die CPU-Auslastung nahe an den kritischen Bereich. Aus diesem Grund sollte ein ständiges Monitoring überprüfen, ob die Performance-Vorgaben eingehalten werden. Die Vorgabe lautet, dass die Antwortzeiten einer Online-Abfrage im Millisekunden-Bereich liegt. Das kann garantiert werden, solange sich die CPU-Auslastung im vorhergesagten Bereich bewegt.

Hinweis

Beachten Sie, dass die hier formulierte Aufgabe eine Kapazitätsabschätzung für OLTP-Abfragen zum Ziel hat. Es sollte unter keinen Umständen zu einer signifikanten Verschlechterung der Antwortzeiten kommen. Wäre es um Batch-Prozesse oder Abfragen in einem Data Warehouse gegangen, dann hätte man einen anderen Ansatz gewählt und andere Ergebnisse erhalten. In dieser Art von Applikationen steht der Gesamtdurchsatz im Vordergrund. Die einzelnen Antwortzeiten sind in der Performance-Betrachtung sekundär.

Dass in diesem Fall ein Performance Monitoring erforderlich ist, zeigt die Tatsache, dass bereits bei einer Verdoppelung der Datenmenge die CPU-Belastung aus dem vorgegebenen Rahmen läuft und sich im kritischen Bereich befindet. Schauen Sie sich die Werte mit 300 Online-Sessions und doppelter Datenmenge in Listing 2.2 an.

Listing 2.2 Workload-Statistiken mit 300 Online-Sessions und doppelter Datenmenge

```
SNAP_TIME           USER_CALLS   ACT_SESS CPU_USER CPU_SYS CPU_TOTAL
------------------- ---------- ---------- -------- ------- ---------
13.02.2008_11:30:09      13876        300    81.35   16.24     97.59
13.02.2008_11:00:05      13645        305    87.38   12.41     99.79
13.02.2008_10:30:06      13459        302    73.16   18.00     91.16
13.02.2008_10:00:02      13295        301    60.11   17.95     78.06
```

Allein eine Verdoppelung der Datenmenge hat dazu geführt, dass die CPU-Auslastung in den kritischen Bereich gestiegen ist. Da es sich in der Spalte *CPU_TOTAL* um einen Durchschnittswert handelt, liegt die Vermutung nahe, dass die CPU zeitweise zu 100 % ausgelastet ist und sich eine *Runqueue* bildet. Die Prozesse stapeln sich in einer Schlange, bevor sie von der CPU abgearbeitet werden können. Dies führt zu erhöhten Antwortzeiten bei den Online-Abfragen. Mit Hilfe des vmstat-Kommandos können Sie die Länge der Runqueue abfragen. Der Wert befindet sich in der ersten Spalte.

Listing 2.3 Abfrage der Runqueue mit vmstat

```
$ vmstat 1 10
procs -----memory---------- ---swap-- -----io---- --system-- -----cpu----
r  b swpd free   buff  cache   si   so    bi    bo   in   cs  us sy id w s
25 0 8  78196   5256 1070016    0    0     0     0 1005  282 85 15  0 0 0
26 0 8  78196   5256 1070016    0    0     0    48 1044  338 81 19  0 0 0
27 0 8  78196   5256 1070016    0    0     0     0 1005  303 80 20  0 0 0
26 0 8  78196   5264 1070008    0    0     0    16 1006  300 85 15  0 0 0
37 0 8  78196   5264 1070008    0    0     0    48 1025  322 84 16  0 0 0
26 0 8  78196   5264 1070016    0    0     0     0 1022  327 91  9  0 0 0
```

Das Beispiel unterstreicht die Bedeutung des Performance Monitoring. In der Praxis sieht das häufig so aus, dass die Datenbank unbeobachtet wächst und vorerst keine Auswirkungen auf die Antwortzeiten der Applikation zu spüren sind. Dann tritt der so genannte *Kipp-Effekt* ein. Die Applikation kippt quasi über Nacht, und es kommt zu einer signifikanten

Verschlechterung der Performance. Besonders gefährdet sind Anwendungen mit vielen Full Table Scans oder Full Index Scans. Ist die Abarbeitung der SQL-Anweisungen dann zusätzlich mit einem hohen Anteil von physikalischen I/O-Operationen verbunden, fällt der Verstärkungsfaktor des Effekts wesentlich größer aus.

Das Monitoring erkennt den steigenden CPU-Bedarf bei einer gleich bleibenden Anzahl von Sessions und sendet rechtzeitig Alerts aus. Damit hat der Administrator noch einen zeitlichen Spielraum, die Ursache zu erkennen und zu reagieren, bevor das Problem entsteht. Wenn sich die Anwender über eine verschlechterte Performance beschweren, ist es in der Regel zu spät, und die Problembeseitigung wird wesentlich teurer.

Woran erkennt der Administrator, dass die Ursache im Wachstum der Datenbank liegt? Im vorliegenden Beispiel ist es relativ einfach, wohingegen es in der Praxis immer komplexer ist. Ein mehrstufiges Monitoring und die Auswertung von AWR-Reports der Datenbank oder die Empfehlungen des Automatic Database Diagnostic Monitor (ADDM) helfen bei der Ursachenfindung.

Unter mehrstufigem Monitoring versteht man die Überwachung auf mehreren Ebenen der Datenbank- und System-Architektur. Die Anzahl von User Calls gibt in diesem Zusammenhang wenig Rückschluss auf das Wachstum der Datenbank. Eine Wachstumsstatistik und eine Statistik über die Anzahl der Buffer Gets führen die Ursachenforschung in die richtige Richtung. Da, wie wir in diesem Fall wissen, die Full Table Scans sehr CPU-lastig sind, hat die Anzahl der gelesenen Buffer einen bedeutenden Einfluss auf den CPU-Verbrauch. Die Anzahl der gelesenen Buffer können Sie aus den System-Statistiken auslesen.

Listing 2.4 Buffer Gets in Workload-Statistiken sammeln

```
#!/bin/ksh
# Sammeln von Workload-Statistiken - buffer gets
export ORACLE_HOME=/opt/oracle/product/11.1.0/db_1
export ORACLE_SID=hanser
export ORACLE_BASE=/opt/oracle
export PATH=$ORACLE_HOME/bin:$PATH
while true; do
sar 9 1 > /tmp/02_01.sar
USR_CPU=`cat /tmp/02_01.sar | tail -1 | awk '{ print $3 }'`
SYS_CPU=`cat /tmp/02_01.sar | tail -1 | awk '{ print $5 }'`
TOT_CPU=`cat /tmp/02_01.sar | tail -1 | awk '{ print $8 }'`
let TOT_CPU=${USR_CPU}+${SYS_CPU}
$ORACLE_HOME/bin/sqlplus -s / as sysdba <<EOF
INSERT INTO pf_simple2 VALUES (seq_simple.nextval,sysdate,(SELECT a.value
+ b.value FROM v\$sysstat a, v\$sysstat b WHERE a.name = 'db block gets'
AND b.name = 'consistent gets'),(SELECT count(*) FROM v\$session WHERE
status = 'ACTIVE' and username IS NOT NULL),$USR_CPU,$SYS_CPU,$TOT_CPU);
COMMIT;
EXIT;
EOF
done
```

Hinweis

Beachten Sie, dass die in den V$-Views gespeicherten System-Statistiken der Datenbank kumulative Werte sind. Um die Durchschnittswerte für den Zeitraum zwischen zwei Snapshots zu erhalten, muss die Differenz aus den beiden Snapshot-Werten gebildet werden. Bei einem Neustart der Datenbank werden die Werte auf Null gesetzt.

Listing 2.5 beinhaltet einen Report der Buffer Gets mit der ursprünglichen Datenmenge in einem Zeitraum von jeweils ca. 10 Sekunden.

Listing 2.5 Report der Buffer Gets mit der ursprünglichen Datenmenge

```
SNAP_TIME           BUFFER GETS CPU_TOTAL
------------------- ----------- ---------
13.02.2008_22:52:24      141090     44,77
13.02.2008_22:52:14      141053     47,68
13.02.2008_22:52:03      141053     46,41
13.02.2008_22:51:53      141097     45,86
13.02.2008_22:51:42      141119     45,50
13.02.2008_22:51:32      141097     45,20
13.02.2008_22:51:21      141075     46,21
13.02.2008_22:51:10      141091     46,86
13.02.2008_22:51:00      147548     61,42
13.02.2008_22:50:49      141053     46,21
```

Vergleichen Sie die Werte mit Listing 2.6, das eine verdoppelte Datenmenge bei gleicher Anzahl von Sessions repräsentiert. Während für das Bilden des Forecasts die Anzahl der aktiven Online-Sessions als wichtigste Statistik gewählt wurde, ist sie durch die Deckelung für das Performance Monitoring eher uninteressant. Die Anzahl der Buffer Gets hingegen ist eine wichtige Größe, da sie den Einfluss des Daten-Wachstums auf die Performance am besten widerspiegelt.

Wichtig

Das Beispiel hat gezeigt, dass die benötigten Statistiken für Forecast und Monitoring nicht identisch sein müssen. Wählen Sie deshalb die Daten und Kategorien gezielt für den jeweiligen Zweck aus. Das gilt auch für das Festlegen der Granularität und der Snapshot-Instervalle.

Listing 2.6 Report der Buffer Gets mit doppelter Datenmenge

```
SNAP_TIME           BUFFER GETS CPU_TOTAL
------------------- ----------- ---------
13.02.2008_23:01:23      301440     97,81
13.02.2008_23:01:13      542464    100,00
13.02.2008_23:01:03      575052     99,90
13.02.2008_23:00:52      282151     94,12
13.02.2008_23:00:42      356400     86,05
13.02.2008_23:00:32      377056     97,37
13.02.2008_22:59:21      434819     83,69
13.02.2008_22:59:11      232197     82,53
13.02.2008_22:59:01      283794     87,35
13.02.2008_22:58:51      283780     86,51
```

Allein aus dem Blickwinkel der Anzahl von Sessions und der ausgeführten SQL-Befehle hat keine Vergrößerung der Workloads stattgefunden. Wenn Sie sich jedoch etwas tiefer in die Datenbank-Architektur begeben und die Anzahl der Buffer Gets betrachten, dann hat eine Verdoppelung der Workloads stattgefunden. Welche Statistiken sollen also für das Performance Monitoring herangezogen und wie soll der Begriff *Workload* definiert werden?

Behalten Sie immer das Ziel im Auge, eine Verschlechterung der Performance nicht zuzulassen und die Ursachen für aktuelle oder zukünftige Veränderungen frühzeitig zu erkennen. Betrachten Sie mit diesem Hintergrund Workload als die Menge aller beachtenswer-

ten Einflüsse auf die Performance, die im Bereich des Capacity Managements liegen. Da das Workload Repository den Workload in konkreten Zahlen widerspiegelt, müssen folgerichtig die Statistiken, die diese Einflüsse am besten charakterisieren, ins Monitoring aufgenommen werden. Zwar gibt es Statistiken, die häufiger benötigt werden als andere, dennoch lässt sich keine allgemeine Empfehlung für die Auswahl von Statistiken geben. Denken Sie immer daran, dass die Zusatzbelastung für die Target-Datenbank so gering wie möglich gehalten werden soll. Eine problemorientierte Auswahl ist wesentlich effektiver.

Ein Forecast-Modell stellt wie jedes andere Modell eine Vereinfachung der äußerst komplexen Realität dar. Bei der Modell-Definition werden bestimmte Einfluss-Faktoren vernachlässigt oder vereinfacht dargestellt. Das hat natürlich Konsequenzen für die Genauigkeit und die Zuverlässigkeit des Forecasts. Sie sollten sich deshalb bei jedem Modell der möglichen Fehlerquellen bewusst sein.

Wenn Sie das Modell im vorliegenden Beispiel analysieren, werden Ihnen einige Vereinfachungen oder Fehlerquellen auffallen. So wird beim Sammeln der Statistiken die durchschnittliche CPU-Belastung über 9 Sekunden ermittelt. Die reale Spitze kann höher liegen. Allerdings können wir diese Fehlerquelle vernachlässigen, da noch ein Puffer nach oben existiert. Eine höhere Snapshot-Frequenz als 10 Sekunden würde zu einer beachtlichen Mehrbelastung der Target-Datenbank führen. Das Modell stellt eine starke Vereinfachung der Realität dar, da es mit gemittelten Werten arbeitet. Im realen Betrieb ist es durchaus möglich, dass die SQL-Abfragen in ihrer Zeitverteilung überlagert werden und es damit temporär zu Warteschlangen kommt. Das würde eine stärkere Abweichung der Maximalwerte vom gebildeten Mittelwert zur Folge haben. Allerdings kann man nach den Gesetzen der Wahrscheinlichkeitstheorie eher von einer seltenen Überlagerung ausgehen. Aus praktischer Sicht stellen selten auftretende Warteschlangen kein Problem dar.

Alles in allem lässt sich festhalten, dass für den vorliegenden Fall und mit den Erkenntnissen über den Charakter der Datenbank und der SQL-Abfragen das Modell und die Art und Weise der gesammelten Statistik-Daten hinreichend zuverlässige Ergebnisse für die Beantwortung der gestellten Frage liefern.

Hinweis

Wenn Sie in einem Modell und für das Sammeln von Statistiken viele mögliche Fehlerquellen und Vereinfachungen identifizieren, dann sind Sie gezwungen, einen größeren Sicherheitspuffer in den Forecast einzubauen. Stellen Sie dabei eine Kosten-/Nutzen-Überlegung an, und überprüfen Sie, ob der Mehraufwand in der Forecast-Ermittlung die Effekte durch die Reduzierung des Sicherheitspuffers aufwiegt. Eine Verkleinerung des Sicherheitspuffers kann Einsparungen bei den Hardware-Kosten oder, wie im vorliegenden Beispiel, zusätzliche Einnahmen durch die Aufschaltung von mehr Benutzern zur Folge haben.

Die Beispiele haben gezeigt, worum es bei der Erstellung von Forecasts geht. Die eingesetzte Methode besteht in einer einfachen statistischen Auswertung der Stichprobe. Es wurde der Mittelwert berechnet und einfach auf die Vorhersage projiziert. Das Problem mit dieser Methode liegt darin, dass implizit zwar Annahmen getroffen, jedoch nicht hinterfragt oder nachgewiesen werden. Die Folge ist eine hoher Unsicherheit bezüglich der

Genauigkeit des Forecasts. Aus diesem Grund muss ein großer Sicherheitspuffer in die Vorhersage eingebaut werden. Die folgenden Voraussetzungen wurden als erfüllt angesehen, jedoch nicht verifiziert:

- Linearität der Abhängigkeiten, bis in den Forecast-Bereich hinein
- Zufälligkeit und Signifikanz der Stichprobe
- Normalverteilung des Fehlerhistogramms
- Symmetrie der Verteilung der Statistik-Werte

Sie werden im Weiteren verschiedene Forecast-Modelle kennenlernen, die eine höhere Zuverlässigkeit und Vorhersage-Genauigkeit garantieren. Die Modelle unterscheiden sich stark in Ansatz und Vorgehen. Die Frage, welches Modell am besten geeignet ist, hängt von der Zielstellung ab.

2.5 Forecast-Modelle

Im vorangegangenen Beispiel wurden Workload-Statistiken gesammelt und ein einfaches Forecast-Modell für die Abhängigkeit des CPU-Verbrauchs von der Anzahl der aktiven Online-Sessions aufgestellt. Der Nachteil des Modells liegt in der großen Unsicherheit über die Genauigkeit der Vorhersage. Es wurden Annahmen getroffen und nicht verifiziert. Im Beispiel erfolgte das Sammeln der Statistiken zur Hauptbelastungszeit des Systems. Auf Basis der Scatterplot-Grafik wurde die Annahme getroffen, dass eine lineare Abhängigkeit besteht. Da sich die Anzahl der Sessions ausschließlich im oberen Bereich befindet, birgt diese Annahme eine gewisse Fehlerquelle für das Forecast-Modell. Die Absicherung gegen diese Ungenauigkeit wurde durch das Vorhalten einer Puffer-Reserve ausgeglichen.

Im Bereich der mathematischen Statistik und Wahrscheinlichkeitsrechnung gibt es eine Reihe von Methoden, die dieses Manko ausgleichen. Durch eine Regressions-Analyse lässt sich die Qualität der Stichprobe mit statistischen Methoden verifizieren. Auch die lineare Abhängigkeit der gewählten Variablen kann nachgewiesen werden. Diese Methoden fallen in den Bereich der mathematischen Forecast-Modelle, die in vielen anderen Bereichen erfolgreich eingesetzt werden.

Benchmark-Modelle liefern Ergebnisse von Prozessen, die auf realen Computer-Systemen und realen Datenbanken laufen. Sie erfassen damit die Komplexität einer Oracle-Datenbank wesentlich besser und liefern reale Ergebnisse. Der Nachteil von Benchmark-Modellen besteht darin, dass sie aufwendig vorzubereiten und durchzuführen sind. Es braucht wesentlich mehr Zeit, einen Benchmark durchzuführen, als eine Vorhersage mit mathematischen Methoden zu treffen.

Für große Datenbanken ist es wirtschaftlich sehr aufwendig, Testsysteme in derselben Größenordnung bereitzustellen. In diesen Fällen bietet sich die Möglichkeit, Benchmarks auf kleineren Systemen durchzuführen und auf ein großes System zu skalieren. Hierfür existieren erprobte Skalierungsmethoden. Dieses Vorgehen ist aus wirtschaftlicher Sicht sehr interessant. Darüber hinaus lässt sich durch die Skalierung eines realen Benchmarks

eine große Vorhersage-Genauigkeit erzielen. Für die Systemeinführung von Applikationen in die Produktion spart diese Methode nicht nur Kosten, sondern auch Zeit.

Die dritte Gruppe von Forecast-Modellen basiert auf der *Queuing-Theorie*. Sie spiegelt die realen Prozesse in einem Computer-System sehr gut wider. Die Zuverlässigkeit dieser Methode hat sich in vielen anderen Bereichen bestätigt. Sie wird zum Beispiel für die Prozess-Steuerung in Call Centern eingesetzt. In einem Computer-System bilden Prozesse Warteschlangen an der CPU oder am I/O Subsystem. Das Problem der Run Queue an der CPU haben Sie bereits kennengelernt.

Wenn es zur Schlangenbildung kommt, erhöhen sich die Antwortzeiten. Erhöhte Antwortzeiten bedeuten Performance-Probleme. Jeder Administrator im Produktions-Umfeld kennt Fragen wie diese:

- Wie viel Erhöhung an Workload verkraftet das aktuelle System?
- Wie viele CPUs muss das neue System haben, um für die nächsten drei Jahre eine hinreichend gute Performance der Datenbank-Applikation sicherzustellen?
- Wie lange kann die Datenbank noch wachsen, ohne dass es zu Performance-Problemen kommt.

Für die Beantwortung solcher oder ähnlicher Fragen sind Queuing-Modelle sehr gut geeignet.

Die Auswahl des oder der Modelle sollte immer problemabhängig passieren. Es gibt keine Standard-Empfehlung für das am besten geeignete Modell. Im weiteren Verlauf des Buches werden Sie Modelle aus den folgenden drei Kategorien kennenlernen:

- Mathematische Forecast-Modelle
- Benchmark-Modelle
- Queuing-Modelle

Jedem dieser Modelle haben wir ein eigenes Kapitel gewidmet. Die folgenden Abschnitte vermitteln schon mal einen kurzen Einblick.

2.5.1 Mathematische Forecast-Modelle

Mathematische Forecast-Modelle basieren auf mathematischer Statistik und Wahrscheinlichkeitsrechnung. Diese Modelle werden erfolgreich in anderen Bereichen wie Verkauf und Marketing oder im Aktienhandel eingesetzt. Die Statistiken werden Analyse-Verfahren unterzogen. Damit wird nachgewiesen, dass die Voraussetzungen des Modells erfüllt sind.

Insbesondere lineare Regressions-Modelle haben sich in der Praxis bewährt. Dabei geht es, wie der Name schon sagt, um lineare Abhängigkeiten. Sie können überall dort eingesetzt werden, wo lineare Abhängigkeiten vermutet werden und die Projektion auf den vorherzusagenden Bereich die Linearität nicht verletzt. Die Qualität der Stichprobe wird mit statistischen Methoden untersucht und bewertet. Es lässt sich zweifelsfrei nachweisen, ob eine lineare Abhängigkeit vorliegt oder nicht.

Betrachten Sie das in diesem Kapitel eingeführte Beispiel und die Grafik in Abbildung 2.5 Das Chart zeigt die für den Forecast verwendete Stichprobe. Allein durch das Betrachten der Grafik kommen Zweifel auf, ob tatsächlich eine lineare Abhängigkeit besteht. Durch eine Regressions-Analyse würde diese Stichprobe glatt durchfallen und das Prädikat „ungeeignet“ für ein lineares Regressions-Modell erhalten.

Eine Statistik, die über einen längeren Zeitraum läuft und eine größere Breite an unabhängigen Werten anbietet, bildet eine bessere Basis für einen Forecast mit dem Regressions-Verfahren. Diese Voraussetzung erfüllt die folgende Statistik:

Listing 2.7 Statistik über eine größere Datenbasis (Auszug)

```
SNAP_TIME           USER_CALLS   ACT_SESS CPU_USER CPU_SYS CPU_TOTAL
------------------ ---------- ---------- -------- ------- ---------
16.02.2008_20:07:17      34907        310    55,56   22,56     78,12
16.02.2008_20:07:07      34898        310    55,45   22,84     78,29
16.02.2008_20:06:58      34889        310    49,44   20,00     69,44
16.02.2008_20:06:49      34880        310    51,05   22,44     73,49
16.02.2008_20:06:38      34865        300    51,23   21,46     72,69
16.02.2008_20:06:27      34856        300    51,23   20,88     72,11
16.02.2008_20:06:16      34847        300    49,54   21,74     71,28
16.02.2008_20:06:03      34838        300    48,54   20,15     68,69
16.02.2008_20:05:52      34823        290    50,52   20,50     71,02
16.02.2008_20:05:42      34814        290    47,97   20,26     68,23
16.02.2008_20:05:32      34805        290    49,57   20,94     70,51
16.02.2008_20:05:22      34796        290    49,42   20,49     69,91
. . .
16.02.2008_19:47:44      33598         20     2,56    0,89      3,45
16.02.2008_19:47:35      33589         20     2,67    0,89      3,56
16.02.2008_19:47:26      33580         20     0,67    0,11      0,78
16.02.2008_19:47:17      33571         20     3,78    2,56      6,34
16.02.2008_19:47:08      33543         10     1,11    0,78      1,89
16.02.2008_19:46:59      33508         10     0,89    0,44      1,33
16.02.2008_19:46:50      33484         10     1,33    1,00      2,33
16.02.2008_19:46:41      33475         10     0,67    0,11      0,78
```

Die Stichprobe weist eine wesentlich breitere Datenbasis aus in einem Bereich von 10 bis 310 Sessions. Das zugehörige Chart in Abbildung 2.9 erweckt rein optisch den Eindruck einer linearen Relation über einen breiten Bereich. Ziel der linearen Regressions-Methode ist es, eine gerade Linie durch die Punkte zu ziehen und damit die lineare Relation durch eine Formel auszudrücken. Für eine einfache lineare Abhängigkeit sieht die Formel wie folgt aus:

$$Y = a + bX + e \ .$$

X ist die unabhängige Variable und repräsentiert die Anzahl von Sessions. Dagegen ist Y die von X abhängige Variable und stellt die CPU-Auslastung in Prozent dar. Die Variable *e* ist der Fehler oder die Abweichung.

Mit Hilfe der gefundenen Formel wird die Abhängigkeit auf den Vorhersage-Bereich projiziert. Doch welche der vielen möglichen Linien eignet sich am besten? Es gibt verschiedene Methoden, um die „beste“ Linie zu bestimmen; die bekannteste von ihnen ist die *Methode der kleinsten Fehlerquadrate*.

Ein weiterer Vorteil der linearen Regressions-Methode besteht darin, dass die Vorhersage-Ergebnisse durch so genannte *Konfidenz-Intervalle* mit konkreten Zahlen bewertet werden

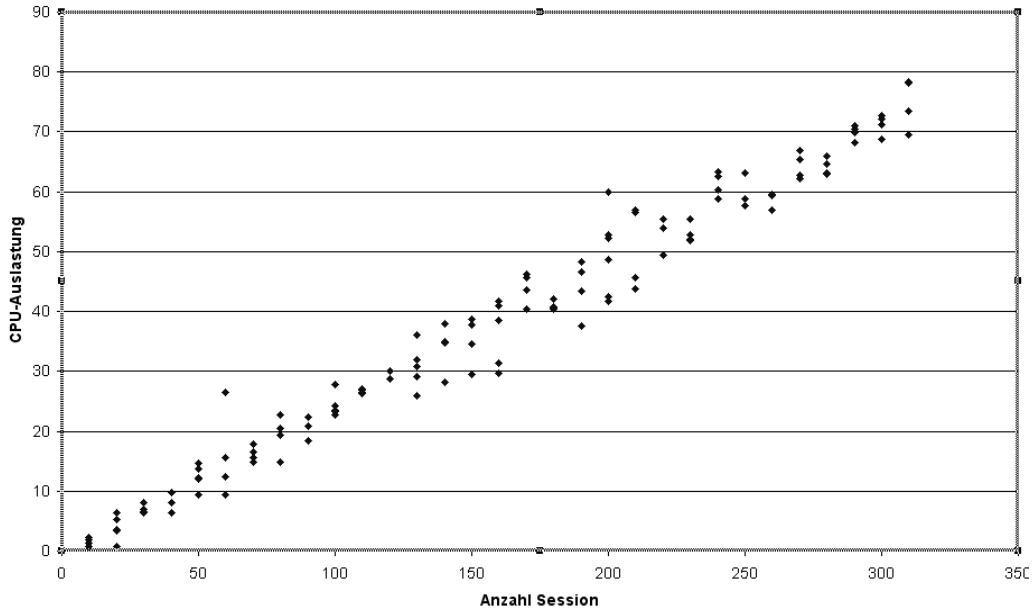

Abbildung 2.9 Scatterplot-Grafik der verbesserten Statistik

können. Ein Konfidenz-Intervall drückt die Wahrscheinlichkeit aus, mit der die Vorhersage-Werte in einem bestimmten Bereich liegen. So können Sie im Forecast-Bericht festhalten, dass die CPU-Auslastung mit einer Wahrscheinlichkeit von 95% in einem bestimmten Intervall liegt.

Sie werden, wenn Sie sich später mit der Queuing-Theorie beschäftigen, erkennen, dass die Abhängigkeit in diesem Beispiel nicht wirklich linear ist. Allerdings ist die Abweichung so klein, dass dieser Fehler vernachlässigbar ist. Sie kann durch die Konfidenz-Intervalle gut ausgedrückt werden.

Lineare Regressions-Modelle zeichnen sich durch eine hohe Vorhersage-Genauigkeit aus. Sie haben jedoch einen wunden Punkt genau dann, wenn der lineare Bereich verlassen wird. Eine Projektion der gefundenen Formel in diesen Bereich wäre äußerst gefährlich.

Für das vorliegende Beispiel ist bekannt, dass die lineare Abhängigkeit nicht mehr gegeben ist, wenn die Auslastung der CPU über 75 Prozent steigt. Das ist in etwa die Grenze, an der sich eine Run Queue bildet. Die Statistik in Abbildung 2.10 bestätigt dies.

Wie Sie noch feststellen werden, ist für diese Art von Forecasts die Queuing-Theorie besser geeignet, da sie die zunehmenden Antwortzeiten bei wachsender Ressourcen-Auslastung berücksichtigt.

Bestandteil des linearen Regressions-Modells ist die *Regressions-Analyse*. Dabei wird geprüft, ob die Statistik die Voraussetzungen für den Einsatz des Modells erfüllt, und unter anderem eine Fehlergrafik erstellt.

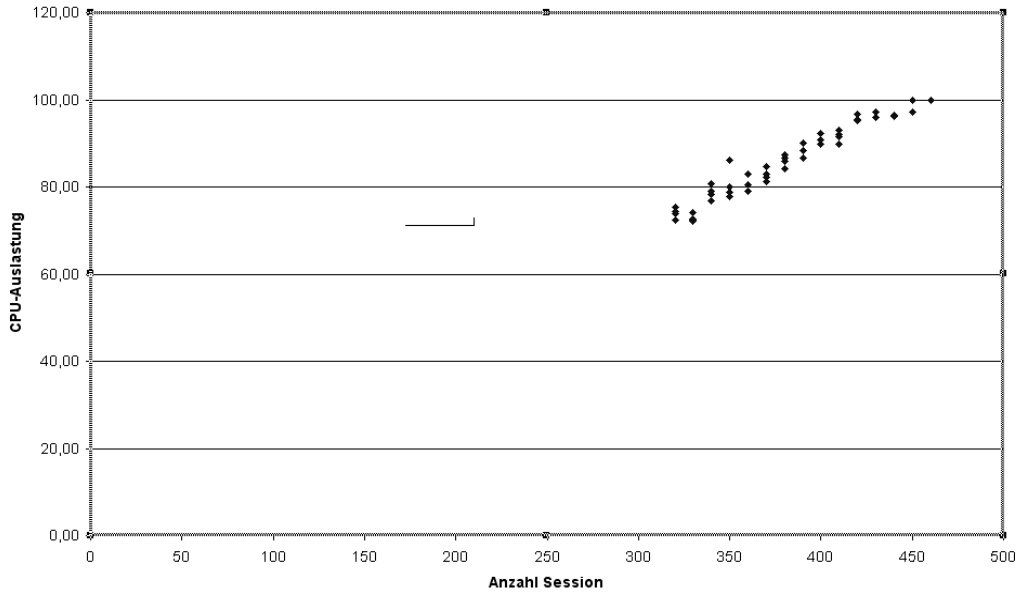

Abbildung 2.10 Nichtlinearer Bereich der Statistik

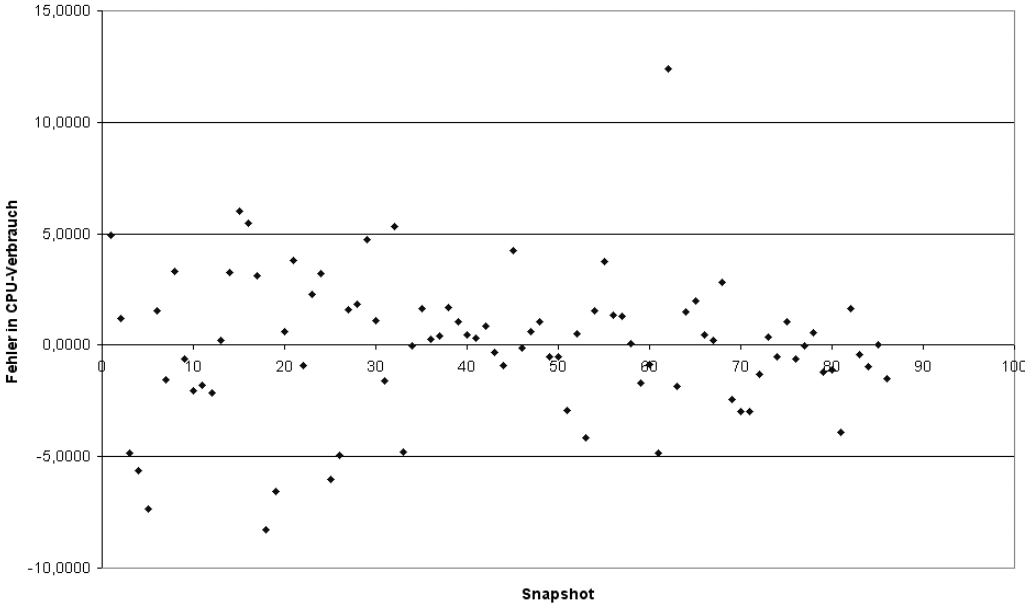

Abbildung 2.11 Fehler-Grafik einer Stichprobe

Durch die Verteilung der Fehler lässt sich erkennen, ob eine lineare Abhängigkeit vorliegt. Wenn das Verteilungsbild wie in Abbildung 2.11 die Form eines Rechtecks annimmt, liegt in der Regel eine lineare Relation vor. Bei anderen Verteilungsbildern kann dies ausge-

schlossen werden. Die endgültige Bestätigung liefert jedoch immer die Regressions-Analyse.

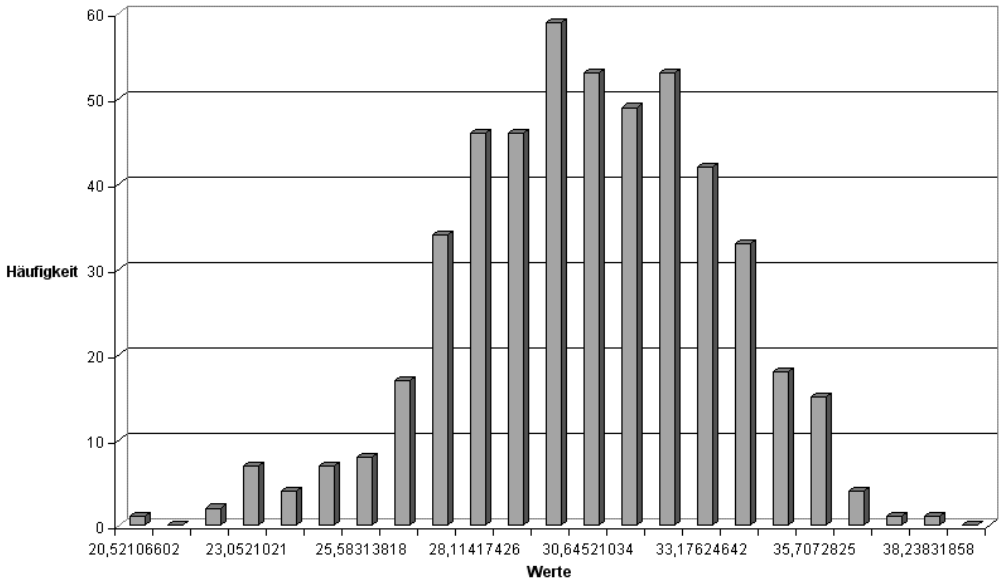

Abbildung 2.12 Fehlerhistogramm einer Stichprobe

Eine Methode, um die Zufälligkeit einer Stichprobe zu prüfen, sind Fehlerhistogramme. Normalität ist gegeben, wenn das Histogramm die Form einer Glockenkurve annimmt. Sie wird deshalb auch als *Normalverteilung* bezeichnet. Auch hier ist noch der mathematische Beweis in Form der Regressions-Analyse anzutreten. Andere Verteilungsbilder lassen in jedem Fall den Umkehrschluss zu, dass die Stichprobe für das Regressions-Modell nicht geeignet ist.

Regressions-Modelle sind nicht auf eine unabhängige Variable beschränkt. Das *Multiple Lineare Regressions-Modell* beschreibt die Relation einer abhängigen von mehreren unabhängigen Variablen. Damit wird das einfache Regressions-Modell zum Spezialfall des multiplen Regressions-Modells. Die Voraussetzungen für beide Modelle sind ähnlich und unterscheiden sich nur durch ihre unterschiedlichen Dimensionen. Die Formel für das multiple Modell lautet:

$$Y_i = b_0 + b_1 X_{1,i} + b_2 X_{2,i} + ... + b_k X_{k,i} + e_i \text{ , für i = 1,2,...,n.}$$

Während für ein einfaches Regressions-Modell die Forecast-Werte über eine Gleichung berechnet werden können, muss für das multiple Modell ein lineares Gleichungssystem gelöst werden, was im Computer-Zeitalter kein Problem darstellt.

Eine detaillierte Darstellung des linearen Regressions-Modells finden Sie in Kapitel 4.

2.5.2 Queuing-Modelle

Das Beispiel im vorangegangenen Abschnitt hat gezeigt, dass ab einem gewissen Auslastungsgrad von Ressourcen eine lineare Abhängigkeit nicht mehr gegeben ist. Dieser Sachverhalt kann durch lineare Regressions-Modelle nicht abgedeckt werden. Die Queuing-Theorie wählt einen anderen Ansatz und spiegelt dieses Verhalten wider.

Queuing-Probleme treten an vielen Stellen im täglichen Leben und in der Volkswirtschaft auf. So ist es nicht verwunderlich, dass sie viele Einsatzgebiete außerhalb der Informationstechnologie gefunden hat. Das klassische Beispiel ist das Call Center. Kunden rufen völlig beliebig und zu unterschiedlichen Zeitpunkten an und müssen mit möglichst geringen Wartezeiten bedient werden.

Es wurde herausgefunden, dass die realen Abläufe in einem Computer durch Queuing-Modelle sehr gut nachempfunden werden können. Auch hat die CPU Anforderungen zu bearbeiten und baut eine Warteschlange auf, wenn mehr Anforderung ankommen, als abgearbeitet werden können.

Die Durchlaufzeit einer Anforderung wird auch Antwortzeit genannt und setzt sich aus der Servicezeit und der Wartezeit in der Schlange zusammen. Die Formel dafür lautet:

$$T_A = T_S + T_W \ .$$

Das ist genau der Grund, weshalb die Kurve im vorangegangenen Beispiel in den nichtlinearen Bereich abgleitet. Je höher der Auslastungsgrad des Systems, desto länger die Warteschlange und desto größer die Antwortzeit. Abbildung 2.13 zeigt einen typischen Kurvenverlauf.

Die Kurve stellt noch einmal grafisch dar, was bereits beschrieben wurde. Ab einem gewissen Auslastungsgrad ist die lineare Regressions-Methode nicht mehr in der Lage, die realen Vorgänge im Modell widerzuspiegeln. Die Queuing-Theorie bildet dagegen die Realität im nichtlinearen Bereich sehr gut ab. Dabei spielt es keine Rolle, ob Sie beispielsweise ein CPU- oder ein I/O-Subsystem betrachten. Der Kurvenverlauf ist immer ähnlich.

Das Queuing-Modell ist deswegen für viele Aufgaben sehr interessant. So lässt sich die Frage beantworten, auf welche Größe eine Datenbank anwachsen kann, bis es zu einer merklichen Zunahme der Antwortzeiten kommt. Sicherlich kennen Sie den so genannten *Kipp-Effekt*. Eine Datenbank läuft über einen längeren Zeitraum problemlos und relativ unbeobachtet. Quasi über Nacht stellen sich Performance-Probleme ein, die so groß sind, dass Ihnen die Anwender die Hölle heiß machen. Diesen Effekt kann man mit der Kurve in Abbildung 2.13 erklären. Sie verläuft über einen großen Bereich sehr flach, ja fast linear, und geht dann plötzlich steil nach oben. Das Queuing-Modell ist in der Lage, diesen Zeitpunkt vorher zu bestimmen und die Katastrophe abzuwenden.

Oder denken Sie an die Planung eines neuen Servers für eine vorhandene Datenbank, die zu kippen droht. Wie viele CPUs soll die neue Hardware haben? Welchen Durchsatz muss das I/O-Subsystem leisten können? Mal ehrlich: Wer ist in Ihrem Bereich in der Lage, diese Fragen möglichst zuverlässig zu beantworten?

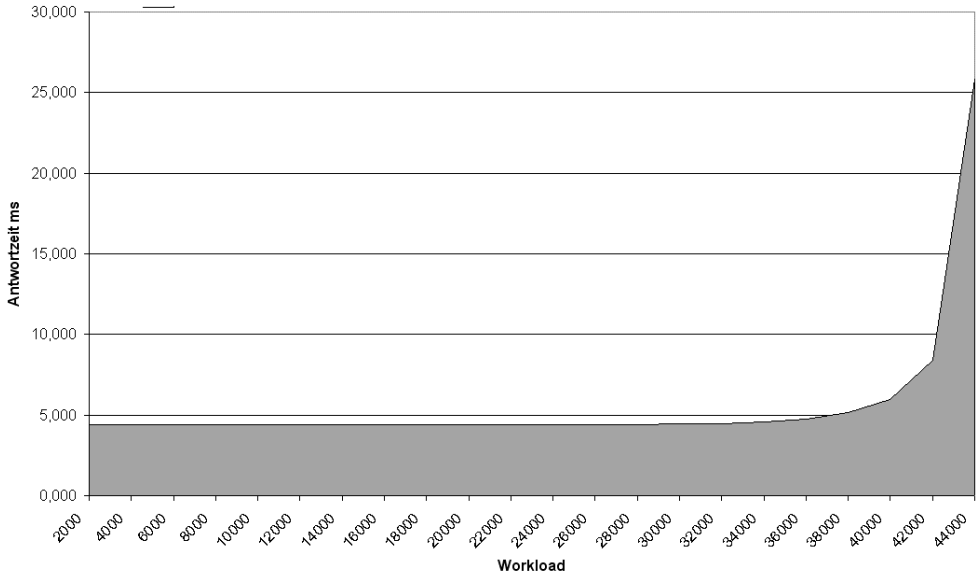

Abbildung 2.13 Verlauf von Service- und Wartezeit nach Auslastungsgrad

In der Regel wird dann geschätzt und ein großer Sicherheitspuffer eingeplant. In der Folge ist die Hardware überdimensioniert, und es wurde zusätzliches IT-Budget verbraten, das man an anderer Stelle dringend benötigt. Auch hier kann ein Forecast mit dem Queuing-Modell zuverlässige Werte liefern, die es Ihnen ermöglichen, einerseits genügend Ressourcen für einen performanten Datenbank-Betrieb bereitzustellen und andererseits keine überflüssigen Ressourcen einzuplanen.

Mit Hilfe der Queuing-Theorie kann ein weiteres Problem gelöst werden. Viele Datenbank-Server sind schlecht ausbalanciert. Das heißt sie stellen entweder zu viele CPU-Kapazitäten im Vergleich zum I/O-Durchsatz bereit oder umgekehrt. Das Chart in Abbildung 2.14 beschreibt eine solche Situation.

Während sich die Antwortzeiten des CPU-Subsystems bei einem Workload von 250% bereits deutlich erhöhen und in den kritischen Bereich übergehen, kann das I/O-Subsystem einen wesentlich größeren Workload verkraften. Auch der umgekehrte Fall ist häufig anzutreffen. Es wurde in Hardware investiert, die nicht benötigt wird.

In Abbildung 2.15 sehen Sie das Ergebnis eines Forecasts mit der Queuing-Methode mit dem Ziel, die optimale Hardware-Ausstattung für einen Datenbank-Server zu finden. Die vertikale Achse zeigt die Antwortzeiten in Abhängigkeit von der CPU-Anzahl.

Mit einer Ausstattung von 3 CPUs liegt die Antwortzeit unter einer Sekunde. Andererseits führt das Hinzunehmen weiterer CPUs nur zu einer minimalen Reduzierung der Antwortzeiten. Damit ist die optimale Konfiguration gefunden. Eine mehr als notwendige Ausstattung lässt sich mit dieser Vorhersage vermeiden.

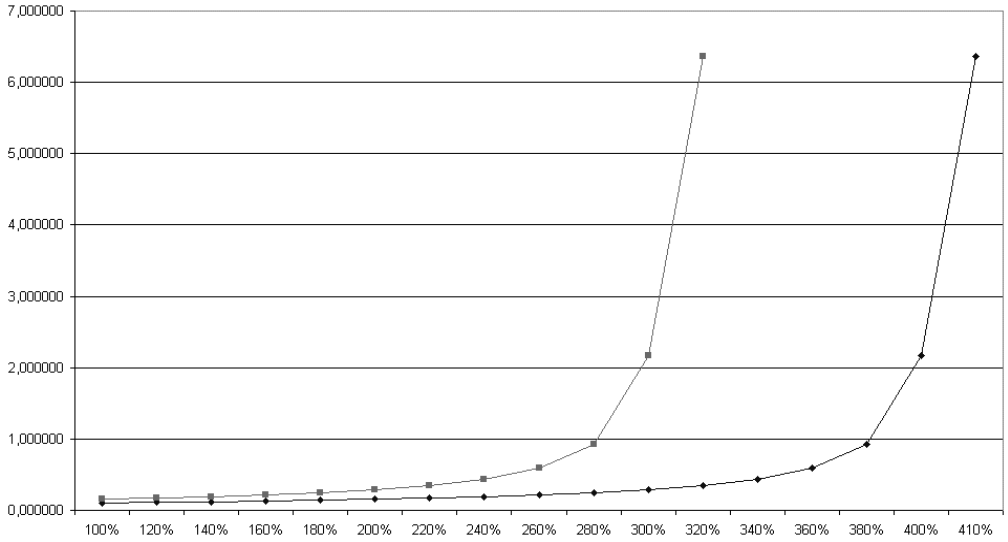

Abbildung 2.14 Auslastung von CPU- und I/O-Subsystem

	A	B	C	D	E	F	G
1	**Antwortzeiten in Abhängigkeit von der Anzahl CPU's**						
2	*Anzahl Queues*	1					
3	*Service-Zeit*	0,0248	*sec/trn*				
4	*Ankunftsrate*	63.984	*trn/sec*				
5							
6	***CPU's***	***Auslastung***	***Service-Zeit***	***Warte-Zeit***	***Antwort-Zeit pro SQL***		
7	2	79,34%	0,0248	0,0421	2,007		
8	3	52,89%	0,0248	0,0047	0,885		
9	4	39,67%	0,0248	0,0009	0,771		
10	5	31,74%	0,0248	0,0001	0,747		
11	6	26,45%	0,0248	0,0001	0,747		

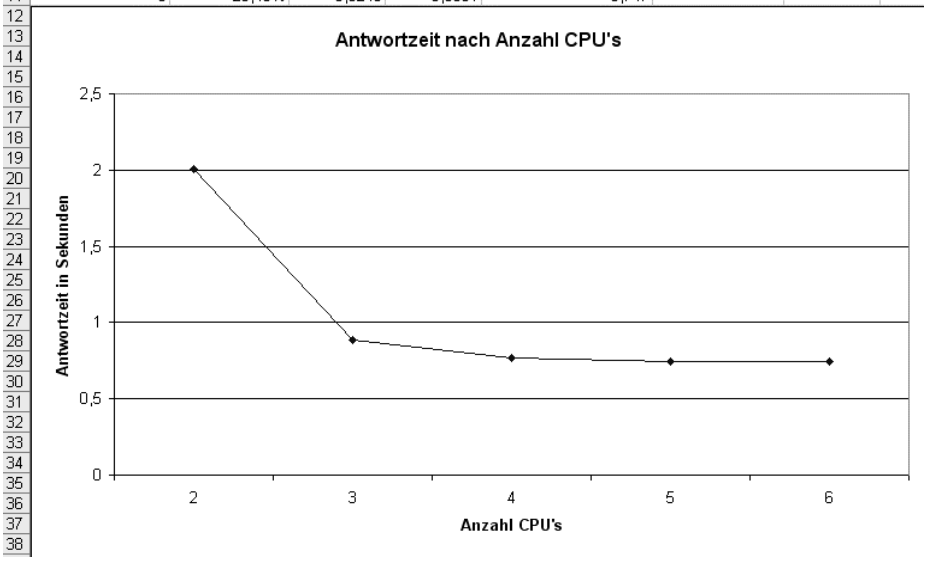

Abbildung 2.15 Ermittlung der benötigten Hardware-Ausstattung

Die Beispiele haben verdeutlicht, welche Ergebnisse mit Queuing-Modellen erzielt werden können. Sie sind besonders gut geeignet, wenn es um Vorhersagen für Antwortzeiten oder den Grad der Komponenten-Auslastung geht. Die Queuing-Methode ermöglicht das Bestimmen der optimalen Hardware-Ausstattung, um weder zu wenige noch zu viele Ressourcen bereitzustellen.

2.5.3 Benchmark-Modelle

Benchmark-Modelle liefern Ergebnisse von Prozessen, die auf realen Computer-Systemen laufen. Darin unterscheiden sie sich grundlegend von den anderen Forecast-Modellen. Sie sind in ihrer Bereitstellung und Durchführung aufwendiger als die mathematischen und die Queuing-Modelle.

Ihre Überlegenheit drückt sich in der Genauigkeit und Zuverlässigkeit der Vorhersagen aus. Schließlich spiegelt ein reales System die Komplexität der Oracle-Datenbank wesentlich besser wider als jedes andere Modell.

Benchmarks verfügen über vielfältige Einsatzgebiete. Sie können mit realem Workload oder mit Workload-Simulationen gefüttert werden. Auf Testsystemen, die in ihrer Ausstattung und Größe identisch sind, kann der Workload unter bestimmten Konditionen getestet werden, bevor eine Übergabe an das produktive System erfolgt. So lassen sich die Auswirkungen einer Erhöhung des Workloads auf dem Testsystem analysieren.

Wenn das Testsystem kleiner als das Produktionssystem ist, dann können die Benchmark-Ergebnisse auf die größere Hardware skaliert werden. Die Skalierung ist ein von mathematischen Verfahren unterstützter Prozess, der die gefundenen Ergebnisse auf die Produktion überträgt. Dieses Verfahren spart aufwendige und teure Tests in angemieteten Rechenzentren und ist sehr gut für Hersteller von Datenbank-Software geeignet, um Vorhersagen für die notwendige Hardware-Ausstattung im Kundenumfeld zu treffen.

Das neue Oracle-Feature *Database Replay* unterstützt diese Prozesse. Der Workload wird auf dem Produktionssystem gesammelt, und die Workload-Dateien werden auf das Testsystem übertragen und dort abgespielt. Der reale Workload aus der Produktion kann für den Benchmark eingesetzt werden, und es muss nicht auf Workload-Simulationen zurückgegriffen werden. Database Replay wird vom Oracle Enterprise Manager unterstützt und kann über eine grafische Benutzeroberfläche komfortabel bedient werden.

Benchmark-Modelle stellen eine sinnvolle Ergänzung dar, wenn sie in Kombination mit anderen Forecast-Modellen eingesetzt werden. Sie tragen dazu bei, die Qualität der Workload-Statistiken zu verbessern, oder können zum Test von Spezialfällen eingesetzt werden.

Um eine Skalierung auf ein größeres System vorzunehmen, wird zuerst die Stichprobe mit statistischen Methoden unter die Lupe genommen. Ähnlich wie bei den linearen Regressions-Modellen wird sie einer Analyse unterzogen, um nachzuweisen, dass die Voraussetzungen des Modells erfüllt sind.

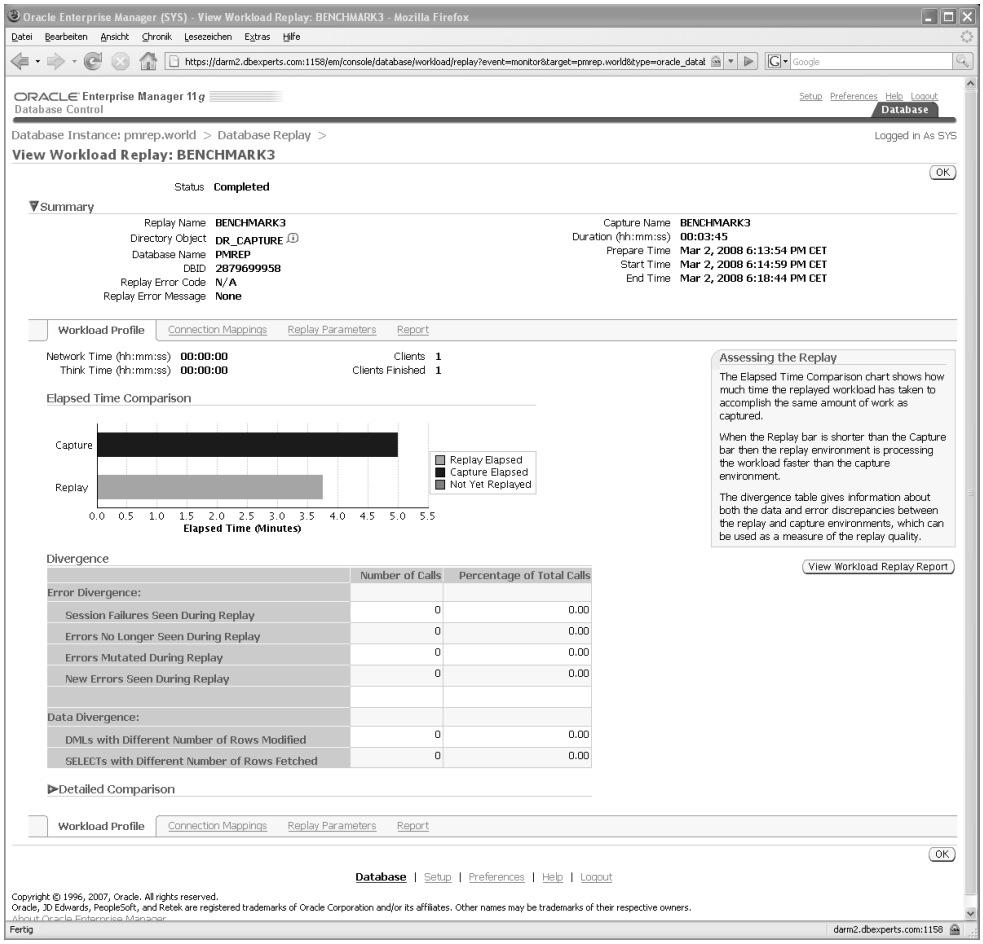

Abbildung 2.16 Database Replay mit dem Enterprise Manager bedienen

Bei der anschließenden Skalierung geht es zum Beispiel darum, die Ergebnisse von 4 auf 16 CPUs zu projizieren. Eine verbreitete Skalierungsmethode ist das *Amdahl'sche Gesetz*, das gut auf die Parallelisierung von Computer-Prozessen angewandt werden kann. Amdahl geht in seinem Ansatz davon aus, dass der Speedup der Parallelisierung durch die seriellen Anteile im Prozess begrenzt wird.

Die Grafik in Abbildung 2.18 zeigt das Verhältnis zwischen physikalischen CPUs und effektiv wirksamen CPUs mit unterschiedlichen seriellen Anteilen von „Null" bis 0,5. Sind die seriellen Anteile gleich „Null", dann ist die Anzahl der effektiven CPUs gleich der Anzahl der realen CPUs. Mit Zunahme des seriellen Anteils wird die Kurve entsprechend flacher.

Das entspricht dem Verhalten, wie es aus der Praxis bekannt ist. Da wissen wir, dass 10 CPUs nicht die zehnfache Leistung einer CPU erbringen.

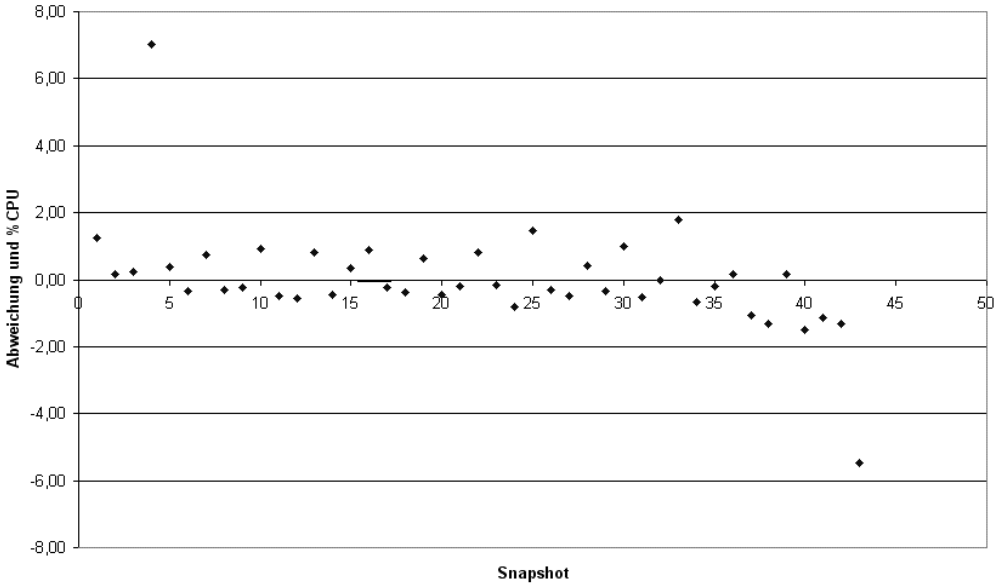

Abbildung 2.17 Fehlerstatistik zur Analyse der Stichprobe

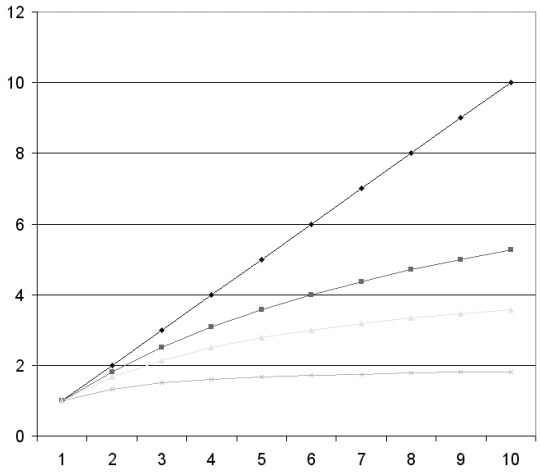

Abbildung 2.18 Das Amdahl'sche Gesetz mit verschiedenen seriellen Anteilen

Obwohl das Amdahl'sche Gesetz – immerhin stammt es aus dem Jahr 1967 – kontrovers diskutiert wird, hat es seine Gültigkeit bewahrt. Ein Amdahl'scher Kritikpunkt ist die Tatsache, dass eine Erhöhung des Parallelisierungsgrades dank der seriellen Anteile im Prozess keinen Sinn mehr ergibt, da der Speedup nur minimal anwächst.

Diese Tatsache ist nach wie vor richtig, wurde aber von *Gustafson* relativiert, der kritisiert, dass Amdahl die Problemgröße konstant hält und diese über die Prozessoren verteilt.

Gustafson nimmt hingegen an, dass in der Praxis die Problemgröße mit der Prozessoranzahl ebenfalls wächst.

Mit anderen Worten: Solange garantiert ist, dass der Workload mit der Leistungsfähigkeit und dem Parallelisierungsgrad Schritt hält, kann mit einer weiteren Parallelisierung immer noch ein Zugewinn an Performance erzielt werden.

Für Datenbanken ist die von *Gunther* entwickelte *Super-Serial-Methode* ein sehr gut geeigneter Skalierungs-Mechanismus. Gunther verfeinert die Formel mit einem zusätzlichen Ansatz. Er sagt, dass sich mit zunehmender Parallelisierung die Concurrency-Probleme verstärken, und führt dafür einen zweiten Faktor in die Formel ein. Das ist genau das Problem, das wir bei Datenbanken häufig feststellen. So steigt mit zunehmendem Parallelisierungsgrad die Wahrscheinlichkeit des Auftretens von Warte-Ereignissen auf Grund von Parallelzugriffen auf dieselbe Ressource. Denken Sie dabei an Buffer Gets oder Latches und Mutexes.

Für Oracle-Datenbanken gibt es Standard- oder Erfahrungswerte für die beiden Parameter der Super-Serial-Methode. Die zugehörige Grafik sehen Sie in Abbildung 2.19. Dies entspricht den praktischen Erfahrungswerten, dass 16 reale CPUs einen Effekt im Bereich von 10 bis 12 erzielen. Für spezielle Applikations-Profile, die bereits einen hohen Anteil an Concurrency mitbringen, kann der Faktor angepasst werden. Die Kurve verläuft dann entsprechend flacher.

Benchmark-Modelle bieten im Zusammenhang mit Skalierungs-Methoden ein breites Einsatzgebiet. So können die Tests vor der Hardware-Bestellung auf kleineren Systemen durchgeführt und die Ergebnisse auf das Produktions-Umfeld skaliert werden. Durch eine Kombination mit Queuing-Modellen lässt sich die Vorhersage-Präzision erhöhen und die optimale Konfiguration bestimmen. Dieses Vorgehen spart nicht nur Hardware-Kosten, sondern verkürzt auch die Zeiten für die Produktionseinführung.

Hersteller von Datenbank-Applikationen haben oft nicht die Testumgebung in der erforderlichen Größenordnung zur Verfügung, um die Performance ihrer Produkte in Produktionsgröße testen zu können.

Durch Benchmarks auf kleinen Testsystemen können sie dennoch den Charakter der Applikation herausfinden und mögliche Skalierungsprobleme erkennen. Probleme im Bereich Concurrency lassen sich auf diese Weise vor der Auslieferung aufdecken und beseitigen. Eine Übertragung der Probleme in die Produktion kann zu einer Vervielfachung führen und die Performance der Applikation in einen kritischen Bereich bringen.

Durch die Skalierung selbst kann das erwartete Verhalten im Produktionsumfeld des einzelnen Kunden bestimmt werden.

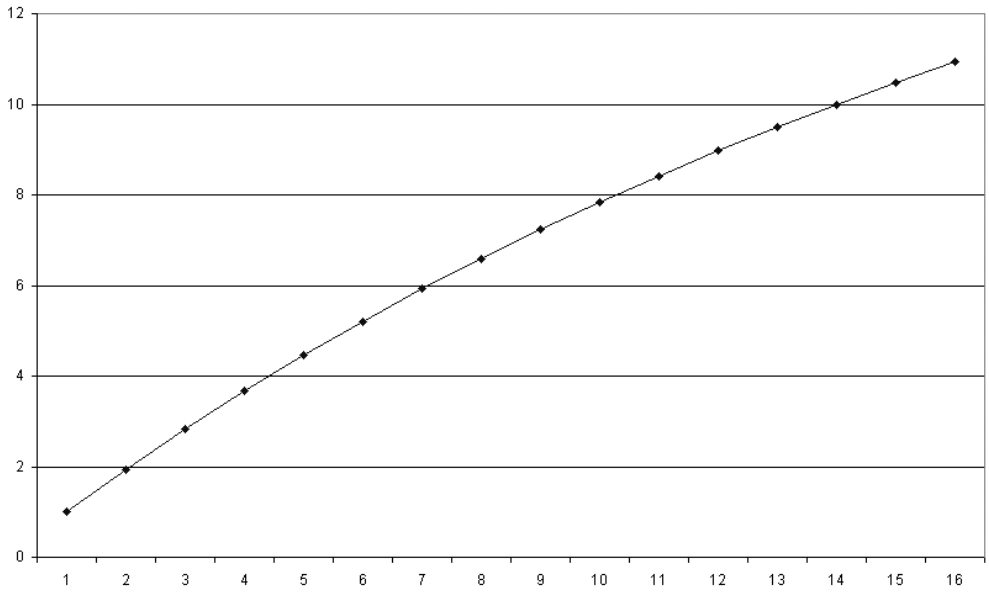

Abbildung 2.19 Skalierung nach Gunther für Oracle-Datenbanken