

---

## 2.1 Begrifflichkeiten

Das Thema Performance-Optimierung gliedert sich in drei Hauptteile:

- System-Performance,
- Anwendungs-Performance und
- Prozess-Performance.

Für alle drei Bereiche existieren wiederum (Abb. 2.1):

- Theorie,
- Messung,
- Analyse und
- Optimierung.

---

## 2.2 Drei Ebenen

Wenn von Performance die Rede ist, wird sehr häufig implizit nur die System-Performance gemeint – oder noch mehr vereinfacht: die Leistungsfähigkeit der Hardware, sprich Prozessor und Hauptspeicher. Das ist mit ein Grund dafür, dass das Thema Performance in den zurück liegenden Jahren vernachlässigt worden ist. Hardware wurde irgendwann

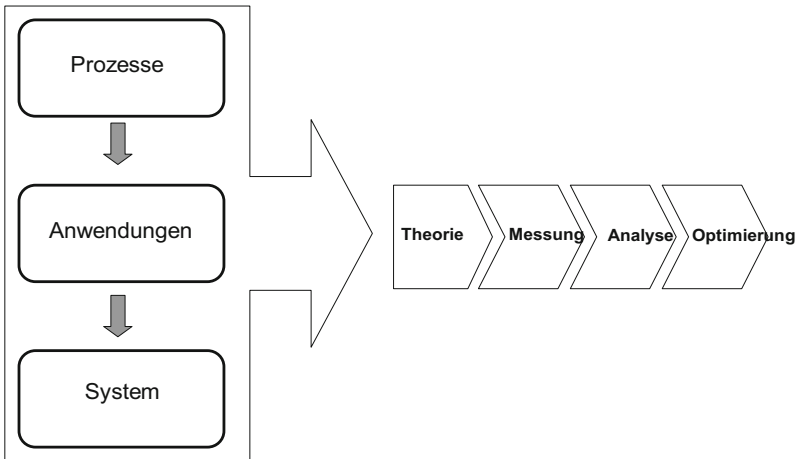


**Abb. 2.1** Performance-Dimensionen

so billig, dass sich programmtechnische Optimierungen nicht mehr zu lohnen schienen, da Manpower eben im Verhältnis zu teuer geworden war. Man kaufte Hardware und Erweiterungen dazu, und schon liefen die Systeme wieder schneller. Oder man war von vornherein so ausgestattet, dass Performance-Probleme einfach nicht auftreten sollten.

Die Enduser-Erfahrungen jedoch sprachen immer schon eine andere Sprache. Nach wie spielt negativ empfundenes Antwortzeitverhalten eine nicht nur psychologisch wichtige Rolle, sondern auch bei der Bewältigung des Durchsatzes im Tagesgeschäft. Das Verhältnis von Hardware-Investitionen zu Optimierungen ist quasi immer konstant geblieben. Die Ursache liegt darin, dass großzügige Hardware-Ressourcen eben so großzügig ausgebeutet werden.

Vor noch vierzig Jahren konnte man sich eine Speicherbelegung mit Leerzeichen oder binären Nullen nicht erlauben. Bereits auf der Ebene der Variablen-Deklarationen und sukzessive bei der Adressierung musste bewusst jedes Byte ausgespart werden. Ansonsten wären Großanwendungen nicht ausführbar gewesen. Spätestens seit Einführung grafischer Oberflächen, mit C++, Java und deren Derivaten war es mit der strukturierten Programmierung im klassischen Verständnis zu Ende. Ansprüche

an Bedienkomfort, Enduser-Queries usw. haben das ihrige getan, um alte Flaschenhälse in neuem Gewand wieder auferstehen zu lassen. Somit ist die Performance-Debatte wieder aktuell geworden – und eben nicht nur auf Systeme und Hardware allein beschränkt. Obwohl zu Anfang (Abb. 2.1) die drei Ebenen:

- System-Performance,
- Anwendungs-Performance,
- Prozess-Performance.

angesprochen wurden, beschränken sich die nun folgenden Begriffsbestimmungen zunächst nur auf System- und Anwendungs-Performance.

Die System-Performance schließt ein (Abb. 2.2):

- Hardware-Auslastung (Speicher, Prozessor),
- Konfiguration der System-Tabellen,
- Ein-/Ausgabe.

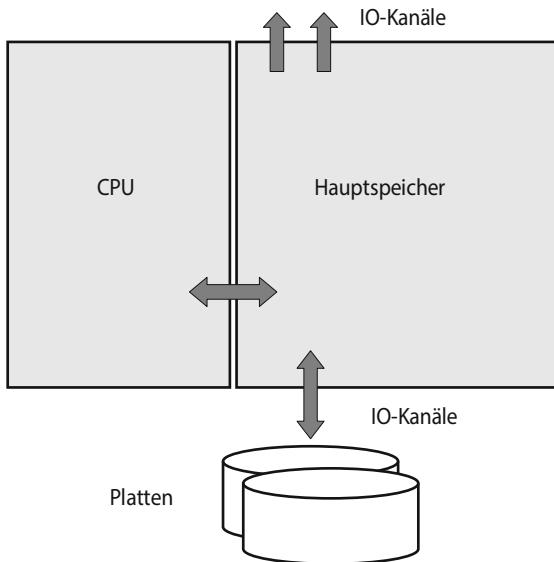


**Abb. 2.2** Untergliederung System-Performance

mit allen für das System-Management relevanten Vorgängen und Parametern.

Bei der Anwendungs-Performance und deren Analyse gibt es natürlich über den Aufruf von System-Ressourcen, die Datenspeicherung und Ein-Ausgabe-Vorgänge mehr oder weniger starke Verquickungen mit den System-Ressourcen im Detail. Insgesamt aber existiert der große Zusammenhang, dass zum Ausführen von Anwendungen eben Systeme benötigt werden. Abb. 2.3 zeigt diesen Gesamtzusammenhang auf. Bei Performance-Betrachtungen können beide Ebenen letztendlich nicht als getrennt nebeneinander existierend betrachtet werden.

Die wesentlichen Elemente, die bei der Anwendungsanalyse eine Rolle spielen sind:

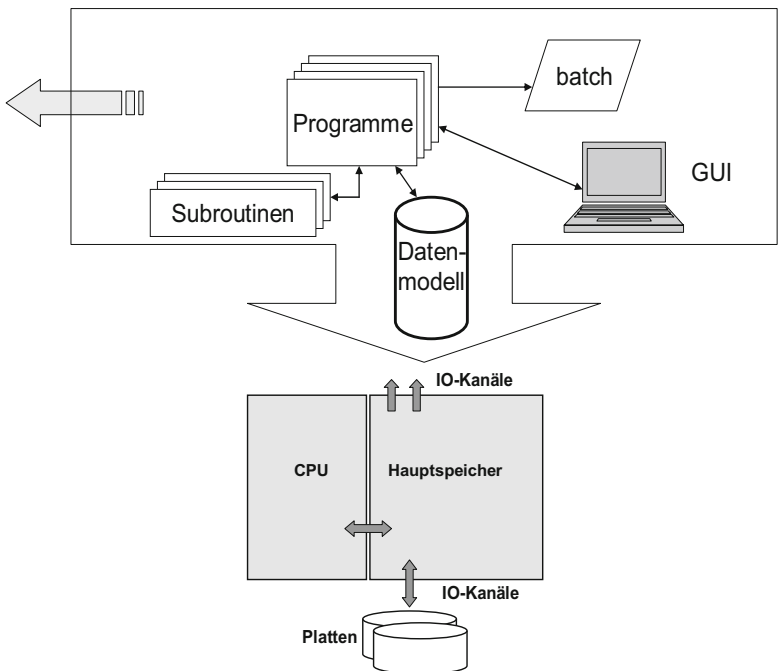


**Abb. 2.3** Zusammenspiel zwischen Anwendungen und System

- Programmstruktur (Gesamtanwendung und Module),
- Datenhaltungskonzept,
- GUI (General User Interface).

Auf der obersten Ebene schließlich steht die Prozess-Performance. Damit ist nicht der systemische Prozessor gemeint, sondern diejenigen Unternehmens-Prozesse, die durch die zu untersuchenden Anwendungen auf ihren Systemen unterstützt werden. Diese Seite der Performance-Debatte wird an dieser Stelle nicht weiter verfolgt. Die Grundsatzfragen sind immer die gleichen:

- Welche System-Stützung ist wirklich erforderlich?
- Wie wird mein Durchsatz durch Anwendungs-Performance beeinflusst?

Auf dieser Basis ist schließlich zu entscheiden, wo beim Tuning am ehesten angesetzt werden muss. An dieser Stelle kommt die Kosten-Nutzen-Frage zu ihrer wahren Bedeutung.

Zusammenfassend lässt sich sagen:

### **Überblick**

Performance-Probleme werden auf der Prozess-Ebene zuerst erkannt, wenn das Tagesgeschäft leidet. An neuralgischen Punkten werden die kritischen Anwendungen identifiziert. Sie wiederum sind abhängig von den Systemen, auf denen sie laufen.

Performance-Messung startet auf der System-Ebene und identifiziert Engpässe, die wiederum auf den Ressourcenverbrauch von Anwendungen zurückzuführen sind.

Soweit die klassischen Ansätze zum Thema Computer-Performance. Wir werden später sehen, dass es weitere und tiefer gehende Möglichkeiten gibt, auf die Performance Einfluss zu nehmen. Doch zunächst noch einige Betrachtungen zur Performance-Theorie.

## 2.3 Performance-Theorie

Wir behandeln die System-Seite und die Anwendungsseite getrennt. Die Praxis lehrt zwar, dass beide Aspekte eigentlich untrennbar miteinander verwoben sind, und Änderungen in der Parametrisierung des einen mit an Sicherheit grenzender Wahrscheinlichkeit Effekte auf der anderen Seite nach sich ziehen werden, aus mindestens zwei Gründen jedoch eine Abschiebung des Gesamtproblems durch eine solche Separatbetrachtung erleichtert wird:

- Auftrennung von Problembereichen nach wechselseitigen Schwerpunkten,
- Identifizierung spezifischer Maßnahmen (ohne mögliche Wechselwirkungen aus den Augen zu verlieren).

Eine vermengte Gesamtbetrachtung der Performance-Komplexität erschwert ungemein das Herausarbeiten von kritischen Einzelaspekten. In diesem Abschnitt werden wir also diese beiden Linien System und Anwendung nacheinander betrachten.

### 2.3.1 System-Performance

Der Abschnitt System-Parameter teilt sich selbst noch einmal auf in

- Hardware-Parameter und
- Betriebssystem-Parameter.

Auch hier haben wir wiederum Verflechtungen, da beide sich bedingen. Beide Elemente machen aus, was man System-Architektur nennt, sollten aber getrennt betrachtet werden, wenn möglich. Das Spektrum möglicher Betriebssystem-Parameter ist natürlich abhängig vom Hardwarehersteller. Auf der anderen Seite ist eine fast unendliche Kombinatorik von Betriebsseinstellungen denkbar für eine gegebene Hardwarekonfiguration desselben Herstellers – nur begrenzt durch die Realitäten der laufenden Anwendungen, wobei wir wieder bei den oben genannten Randbedingungen wären. Umgekehrt kann dieselbe Version eines Betriebssystems auf unterschiedlichen Hardwareumgebungen zuhause sein. Wir werden zunächst auf die Hardware selbst eingehen.

### 2.3.1.1 Hardware-Parameter

#### 2.3.1.1.1 Allgemeine Hinweise

Eigentlich müsste es besser heißen: System- beziehungsweise Hardware-Komponenten. Zu denen, die losgelöst vom Gesamtsystem betrachtet werden können, gehören (Abb. 2.2):

- CPU,
- Hauptspeicher,
- Plattenspeicher,
- Ein-/Ausgabe-Kanäle.

Selbstverständlich gehören zur Hardware noch viele andere Elemente, wie Endgeräte, Modems und andere Kommunikationskomponenten, die aber für unsere Performance-Betrachtungen hier nicht gesondert abgehandelt werden sollen (zum Beispiel: Konfiguration der Cursor-Geschwindigkeit durch Mauseinstellungen). Die Theorie der System-Performance behandelt die oben genannten Komponenten in dieser Reihenfolge. Wir werden uns hier jedoch auf CPU und Hauptspeicher beschränken. Das Gewicht der jeweiligen Komponente bezogen auf eine bestimmte Performance-Situation hängt von der Art der Anwendung, der Anzahl User und weiteren Faktoren ab. Deshalb sollte man im Vorfeld keine Prioritätenfestlegung treffen.

Auch zwischen diesen Komponenten gibt es Beziehungen, die jeweils auch aufgezeigt werden. Obwohl die erwähnten Ressourcen zunächst isoliert betrachtet werden, hängt die Bedeutung für spezifische Performance-Situationen natürlich wiederum – wie bereits erwähnt – von den Anwendungen ab, die diese Ressourcen in Anspruch nehmen. Zur Einkreisung eines spezifischen Problems sind dennoch zunächst Aussagen erforderlich, die grundsätzliche Mechanismen offen legen, die unabhängig von den jeweiligen Anwendungsfällen Gültigkeit besitzen.

Bevor wir uns nun den Ressourcen und deren Eigenheiten im Detail zuwenden, sollen noch einige allgemeine Grundsätze angesprochen werden. Dazu gehören auch die Fragen:

- Wie kann System-Performance grundsätzlich getestet werden? und
- Wann macht Performance-Messung Sinn?

Performance-Testung ist eine Disziplin, über die man auf Menschen, Prozesse und Technologie Einfluss nehmen kann, um Risiken zu vermeiden, die z. B. bei System-Einführung, Upgrades oder Patch-Einspielung entstehen können. Kurz gesagt: Performance-Tests bestehen dann darin, eine typische Produktionssystemlast zu erzeugen, bevor zum Beispiel neue Anwendungen eingespielt werden, um das Leistungsverhalten zu messen, zu analysieren und Enduser-Erfahrung zu sammeln. Ziel ist es also, Performance-Probleme unter Produktionsbedingungen zu identifizieren und zu beheben. Ein gut vorbereiteter Performance-Test sollte in der Lage sein, die folgenden Fragen zu beantworten:

- Ist das Antwortzeitverhalten für die Enduser zufriedenstellend?
- Kann die Anwendung die voraussichtliche System-Last bewältigen?
- Kann die Anwendung die Anzahl Transaktionen, die durch die Geschäftsvorfälle zu erwarten sind, bewältigen?
- Bleibt die Anwendung stabil unter zu erwartenden, aber auch unerwarteten Lastzuständen?
- Werden Enduser beim Scharfschalten positive überrascht sein oder eher nicht?

Indem diese Fragen beantwortet werden, hilft Performance-Testung, die Auswirkungen von Veränderungen und die Risiken von System-Einführungen zu kontrollieren. In diesem Sinne sollten Performance-Tests folgende Aktivitäten beinhalten:

- Emulation von Dutzenden, Hunderten oder gar Tausenden von Endusern, die mit dem System interagieren,
- konsistente Last-Wiederholungen,
- Antwortzeitverhalten messen,
- System-Komponenten unter Last messen,
- Analyse und Abschlussbericht.

Tests und Analyse sollten stattfinden, bevor Engpasssituationen auftreten. Ansonsten müssen später Performance-Messungen am laufenden System durchgeführt und entsprechende Maßnahmen ergriffen werden. Das bedeutet, dass man die Lastverteilung über den Arbeitstag bereits im Vorfeld abschätzen möchte, oder auch, welche Jobs dürfen wann laufen.



**Abb. 2.4** Schritte zur Performance-Optimierung (nach HP Dokument 4AA1-4227ENW)

Finden sich nach Einführung dann tatsächlich Durchsatzprobleme, bietet sich folgende allgemeine Vorgehensreihenfolge an (Abb. 2.4):

- zunächst CPU unter Last messen,
- dann Speicherauslastung,
- Sind CPU und Speicher problemlos, sollten die Ein-Ausgabevorgänge zu den Plattenspeichern untersucht werden.

Neben der bisher diskutierten vorausschauenden Performance-Beobachtung sollten in regelmäßigen Abständen oder kontinuierlich Messungen am laufenden System erfolgen – insbesondere dann, wenn subjektiv der Verdacht schlechter Performance gemeldet wird: lange Antwortzeiten im Dialog, lang laufende Auswertungen.

Ein Problem dabei ist natürlich die anzusetzende Messlatte. Das subjektive Empfinden eines vor einem stummen Bildschirm sitzenden Nutzers, der auf die Rückkehr seines Cursors wartet – also das reine Antwortzeitverhalten – ist meistens nicht ausreichend, um als messbare Größe zu gelten. In der Regel werden deshalb neben den online-Funktionen oder GUIs zusätzlich batch-Läufe, deren Exekutionsdauer sich quantifizieren lässt, herangezogen, um ein Bild über den Gesamtzusammenhang zu bekommen.

Leistungs-Tests stellen also ein Werkzeug dar, mit dem sich feststellen lässt, ob eine System-Anpassung zu eine Verbesserung oder Verschlechterung führen wird. Dabei darf die Konzeptionierung solcher Tests in der

Vorbereitung nicht unterschätzt werden. Sie hängt überdies von den konkreten Verhältnissen vor Ort ab:

- Anzahl User,
- erwarteter Durchsatz,
- vorhandene Hardware usw.

Wichtig ist auf jeden Fall, dass nicht einzelne Anwendungen isoliert getestet werden, sondern reale Bedingungen entweder simuliert oder wie vorgefunden betrachtet werden, bei denen viele System-Prozesse parallel laufen.

Für den einzelnen Anwender reduzieren sich Leistungs-Tests auf ganz handfeste Kennzahlen:

- Antwortzeit und
- Verweilzeit.

Das sind Größen, die direkte Auswirkungen haben auf seinen Arbeitsfortschritt und damit auf seine persönliche Produktivität unter Zuhilfenahme seiner IT-Infrastruktur. Seine Prozess-Schritte setzen sich insgesamt aus der reinen Bedienzeit und einer eventuellen Wartezeit auf System-Antworten bzw. -verfügbarkeiten zusammen. Wartezeiten sind eine direkte Konsequenz der jeweils aktuellen System-Auslastung. Genauso hat also Performance-Tuning einen direkten Effekt auf die Produktivität.

Demgegenüber stehen die Erwartungen des operativen Betriebs, die in eine andere Richtung gehen. Das Interesse hier orientiert sich an der Durchsatzmaximierung auf der Jobebene – also eben auf die Erzeugung hoher Auslastungen. Beide Erwartungshaltungen – die des Anwenders und die des operativen Betriebs – müssen beim Tuning kompromisshaft zusammengeführt werden. Dafür stehen gewisse Richtwerte zur Verfügung, die fallweise unterstützend herangezogen werden können. Dabei handelt es sich nicht allein um quantitativ-technische Werte, sondern um Abwägungen, die insgesamt eine ineffiziente Nutzung von System-Ressourcen verhindern sollen. Vor den ersten Maßnahmen der Performance-Analyse sind folgende Klärungen sinnvoll:

- Geht es um reine Durchsatzprobleme?
- Ist die Performance des Gesamtsystems unzureichend?

Quantitativ lassen sich dazu folgende Eingrenzungen machen:

- Transaktionsraten in Abhängigkeit vom Antwortzeitverhalten,
- Durchsatzraten in Abhängigkeit von Jobverweilzeiten.

Auf den Anwender bezogen gibt es weitere Kenngrößen:

- Antwortzeiten bezogen auf den Transaktionstyp,
- Verweilzeiten für beauftragte Jobs.

Das Leistungsverhalten des Systems lässt sich nun wiederum auf seine einzelnen Komponenten umlegen; andererseits können sich auch organisatorische Schwächen der Produktionssteuerung dahinter verbergen, wenn eine unausgewogene Jobsteuerung dahinter steckt. Beim Einsatz von Mess-Systemen ist zu berücksichtigen, dass solche Systeme ihrerseits auch wieder Ressourcen verbrauchen.

Ein wichtiger Richtwert bei der Messung ist die sogenannte System-Zeit. Sie setzt sich aus folgenden Anteilen zusammen: Zeiten zur

- Steuerung des Timesharing mehrerer gleichzeitiger Prozesse,
- Steuerung von Ein-Ausgaben,
- Steuerung des Swappings von Hauptspeicherbeladungen.

Aus der Erfahrung kann man für die System-Zeit folgende Richtwerte für die CPU festlegen:

- System-Zeit insgesamt: 10–20 %,
- Timesharing: 5–10 %,
- Ein-Ausgaben: 2–6 %,
- Swapping: 1–2 %.

Abb. 2.5 und 2.6 zeigen grundsätzliche Vorgehensweisen im Ablauf eines Tuning-Prozesses. Zum Schluss noch zwei generelle Hinweise: Aufrüstung vorhandener Hardware ist meistens nicht das Allheilmittel

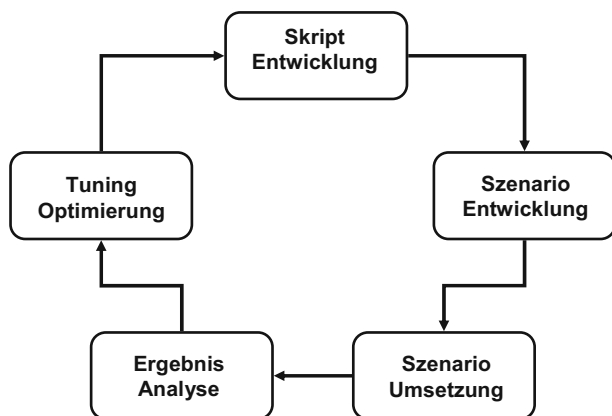


**Abb. 2.5** Optimierungszyklus (nach HP-Dokument 4AA1-5197ENW)

bei Performance-Engpässen, wie weiter unten sichtbar werden wird. Und: Performance-Probleme wachsen exponentiell mit der zu bewältigenden Datenmenge. Nach diesen grundsätzlichen Vorbetrachtungen folgt nunmehr die Einzelbetrachtung der fraglichen Ressourcen.

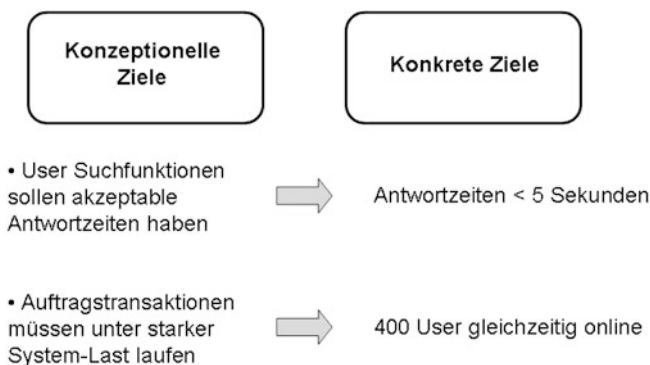


**Abb. 2.6** Von der allgemeinen Konzeption zu konkreten Zielen (nach HP-Dokument 4AA1-5197ENW)

### 2.3.1.1.2 CPU

CPU steht für Central Processor Unit. Unter dem Gesichtspunkt der Performance geht es bei der CPU-Betrachtung um Leistung, d. h. Durchsatz über den Prozessor. Gemessen wird diese Leistung in mips: million instructions per second. Dabei handelt es sich um eine reine Papiergröße, die etwas über nutzbare Leistung aussagt und über Overheads und Task-Verarbeitung ohne Eingabe-/Ausgabe-Vorgänge oder Warteschlangenmanagement für nicht verfügbare andere Ressourcen, auch nicht Virtual Memory Management.

CPU-Leistung wird ein kritischer Faktor für Anwendungen, die stark CPU-gebunden sind: wissenschaftliche und technische Anwendungen mit langen Sequenzen mathematischer Berechnungen.

Eine Messgröße kann dabei der „Relative Performance-Faktor (RPF)“ sein. Dieser Wert macht eine Aussage über die Leistungsfähigkeit einer CPU – und zwar unter möglichst realen Produktionsbedingungen. Zu seiner Ermittlung werden sowohl Online-Anwendungen als auch Batch-Verarbeitung hinzu gezogen. Bei ersteren wird die Transaktionsrate als Maß, bei letzterer die Verarbeitung von Jobs pro Zeiteinheit herangezogen. Je nach Austarierung der Anwendungslandschaft werden diese Maße gewichtet, z. B. 20 % für Batch, 80 % für Online. Mit einer einzigen Zahl kann man allerdings keine schlüssigen Aussagen über das CPU-Verhalten insgesamt machen. In der Regel ist die Anwendungslandschaft dazu zu komplex. Weitere Einflussfaktoren sind durch die Serverarchitekturen gegeben. All diese Überlegungen machen natürlich nur Sinn, wenn CPU-Performance und Hauptspeichergröße sich von vornherein in einem vernünftigen Verhältnis zueinander befinden.

Eine CPU kann sich in folgenden Zuständen befinden:

- user busy,
- Overhead-Verarbeitung,
- im Wartezustand,
- idle.

User busy bezieht sich auf das Ausführen von Anwendungstasks; Overhead-Verarbeitung deutet darauf hin, dass sich die CPU mit sich selbst beschäftigt, z. B. in dem sie durch die Warteschlangen geht oder Prioritäten neu verteilt. Wartezustand weist darauf hin, dass eine benö-

tigte Ressource nicht verfügbar ist, und „idle“ heißt, dass zur Zeit keine Anforderungen an die CPU vorliegen. Die Overheads kann man noch weiter spezifizieren:

- Memory Management / Paging,
- Process Interrupt Control,
- Cache Management.

Ganz anders sieht das Problem in einer Multiprozessoren Umgebung aus (Tab. 2.1). Diese Umgebung konstituiert sich einmal aus den Verarbeitungsprozessoren, andererseits auch aus einer dazu im Verhältnis geringeren Anzahl von I/O-Prozessoren. Um diese Konfigurationen optimal auszunutzen, müssen die Anwendungen entsprechen geschrieben sein. Dazu gehört unter anderem Parallelverarbeitung ohne gegenseitige Interferenzen.

Davon zu unterscheiden sind Konfigurationen, die Prozessoren parallel betreiben unter dem Gesichtspunkt der Ausfallsicherheit, um eine kontinuierliche Verfügbarkeit von Systemen zu gewährleisten (Abb. 2.7).

Auf der Ebene der Tasks kann einer bestimmten Task immer nur ein Prozessor zugeordnet werden und umgekehrt. Daraus ergibt sich schon, dass durch reine Hardware-Aufstockung (mehr Prozessoren) Performance-Probleme nur bedingt gelöst werden können, was das Antwortzeitverhalten betrifft. Im Grenzfall findet lediglich eine schnellere CPU-Zuteilung, aber keine sonstige Durchsatzverbesserung statt. Ebenso ist eine signifikante Beschleunigung von Batchläufen nur durch massive Paralleli-

**Tab. 2.1** Erhöhung Management-Overheads in Multiple CPU-Umgebung (nach Siemens: BS2000/OSD Performance-Handbuch Nov. 2009)

| Anzahl CPUs | Erhöhung CPU-Zeit [%] |
|-------------|-----------------------|
| 2           | 5–10                  |
| 4           | 10–15                 |
| 6           | 15–20                 |
| 8           | 20–30                 |
| 10          | 25–35                 |
| 12          | 30–40                 |
| 15          | 35–45                 |

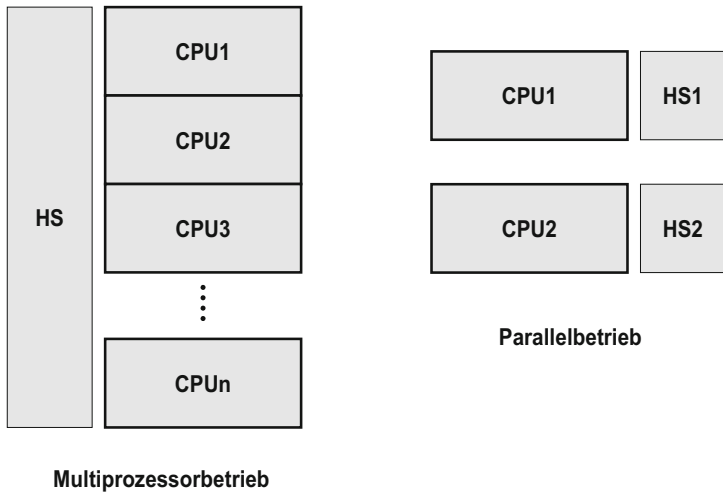


**Abb. 2.7** CPU-Betriebsarten

tät zu erreichen. Daneben ist zu beachten, dass mit steigender Prozessorzahl die Anforderungen an die Synchronisation mit dem Hauptspeicher steigen, und dadurch zusätzliche Overheads erzeugt werden. Die Tab. 2.2 und 2.3 zeigen beispielhaft für Siemenssysteme unter BS2000 die Verbesserungsmöglichkeiten für die Verarbeitung unabhängiger Tasks durch Zuschaltung mehrere CPUs.

**Tab. 2.2** Verbesserungseffekt durch CPU-Zuschaltung (nach Siemens: BS2000/OSD Performance-Handbuch Nov. 2009)

| Anzahl CPUs | Faktor  |
|-------------|---------|
| 2           | 1,7–1,9 |
| 4           | 3,3–3,6 |
| 6           | 4,6–5,2 |
| 8           | 5,8–6,6 |
| 10          | 6,8–7,8 |
| 12          | 7,8–8,8 |
| 15          | 8,8–9,9 |

**Tab. 2.3** Auslastung in Abhängigkeit der Anzahl CPUs (nach Siemens: BS2000/OSD Performance-Handbuch Nov. 2009)

| Auslastung [%] | Anzahl CPUs |
|----------------|-------------|
| 75             | 2           |
| 80             | 4           |
| 85             | 6           |
| 90             | 8           |

Solche Upgrades haben in der Regel zur Folge, dass auch Hauptspeicher und Laufwerke angepasst werden müssen.

Viele Hersteller geben Richtwerte bezüglich der optimalen Auslastung ihrer CPUs. Betrachtet werden im Weiteren lediglich Monoprozessoren. Bei Online-orientierten Systemen sollte die CPU-Auslastung durch die Hauptanwendungen 70 % nicht übersteigen. Darüber hinaus kann es zu Problemen beim Management der Warteschlangen und den zugehörigen Wartezeiten kommen. Naturgemäß gehen die möglichen Auslastungszahlen in Multiprozessorumgebungen über diesen Richtwert hinaus.

Um die Tasks in der CPU zu steuern, werden bestimmte Routinen eingesetzt, um sie zu managen und zu überwachen. Folgende Parameter spielen beim Task-Management eine Rolle:

- Aktivierung,
- Initiierung,
- Multi-Programming Level,
- Priorität,
- Multiprogramming-Level (MPL) pro Kategorie,
- Betriebsmittel-Auslastung (CPU, Hauptspeicher, Paging Aktivität),
- System-Dienste (CPU-Zeit, Anzahl Ein-/Ausgaben),
- Zuteilung der Berechtigung zur Hauptspeicher-Nutzung,
- Zuteilung des Betriebsmittels CPU,
- Zustand „active, ready“,
- Deinitiierung,
- Betriebsmittelentzug bei Wartezeiten,
- Zustand „active, not ready“,
- Verdrängung.

Da die CPU von vielen Tasks gleichzeitig genutzt wird, ist die Auslastung ein wichtiges Kriterium für Wartezeiten. Die durchschnittliche Bedienzeit durch die CPU ist allerdings klein im Verhältnis zu den Ein- und Ausgabzeiten. Das bedeutet, dass das Antwortzeitverhalten wesentlich von Ein-/Ausgabevorgängen, insbesondere durch das Lesen und Schreiben auf Speichermedien beeinflusst wird. Dadurch kommen die bereits genannten 70 % Auslastung zustande. Einige Hersteller erlauben zudem eine manuelle Prioritätenvergabe, sodass auch ursprünglich niedrig priorisierte Tasks aus den Warteschlangen heraus zum Zuge kommen können. Eine 100 %ige Auslastung ist ein idealer Wert, der in der Praxis nicht erreicht wird.

Prioritäten beziehen sich auf den Zugriff auf CPU-Ressourcen, I/Os sind dabei nicht berücksichtigt. Im Normalfall sind Prioritäten so vergeben, dass Online-Taks grundsätzlich höher priorisiert sind als Batches. System-Programme wiederum haben meistens eine höhere Priorität als die übrigen Online-Funktionen. Innerhalb der Programmverarbeitung berechnet das System dann selbst die Prioritäten nach einem Algorithmus, der Warteschlangen-Position, Swap-Rate und andere Parameter einbezieht. Die Vergabe einer externen fixen Priorität ist möglich, sollte jedoch mit Vorsicht gehandhabt werden, da die Priorität sich danach nicht mehr dynamisch anpasst und je nach Einstellung zu Verdrängungen anderer Prozesse führen kann. So kann man grundsätzlich wenigen Nutzern, die große Programme fahren, niedrigere Prioritäten geben als den vielen anderen Usern, die nur kleine Tasks erledigen wollen. Kurzfristig kann man bei eiligen Geschäftsvorfällen die Priorität erhöhen, ebenso für Speicher intensive Programme, die alles blockieren, um das System wieder freizuschaukeln (Abb. 2.8).

Spricht man von einer hohen CPU-Auslastung, dann liegt diese bei über 90 %. Das braucht noch keine Engpasssituation zu sein. Bei hoher Auslastung ist es allerdings besonders wichtig, die Effizienz des CPU-Betriebes zu beobachten. Dazu sind folgende Maßnahmen von Bedeutung:

- Überprüfen der Höhe der Ein-/Ausgaberate,
- Überprüfen der Höhe der Paging-Rate.

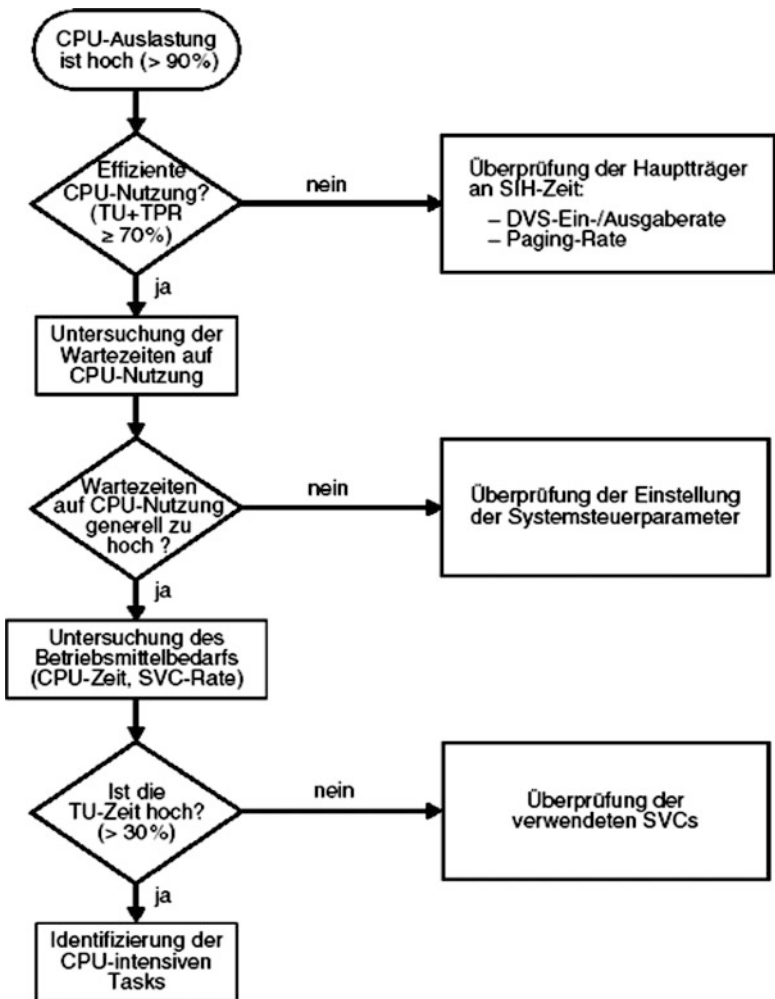


**Abb. 2.8** Tuning Ansätze bei hoher CPU\_Auslastung (nach Siemens: BS2000/OSD Performance-Handbuch Nov. 2009)

Ein Maß für die Effizienz einer CPU bei hoher Auslastung ist das Verhältnis:

Wartezeit zu Benutzung der CPU / Hardware-Bedienzeit der CPU.

Beim Auftreten längerer Wartezeiten sind folgende Ursachen möglich:

- Sub-optimale Einstellung von System-Parametern,
- Gesamter CPU-Zeitbedarf für alle Tasks ist zu hoch. Eine Analyse der Warteschlangenzeiten kann darüber Aufschluss geben.

Eine wichtige Rolle beim Mechanismus der CPU-Verwaltung ist der Umgang mit Unterbrechungen. Offensichtlich finden beim Swapping Unterbrechungen von exekutierendem Code statt. Dadurch werden ein Programm oder ein Programmteil angehalten, das Code-Segment aus dem Hauptspeicher ausgeladen und die Task in eine Warteschlange verwiesen, aus der sie entsprechend ihrer Priorität (im Normalfall: Position in der Schlange) später wieder Ressourcen bekommt. Unterbrechungen können folgende Ursachen haben:

- prioritärer Ressourcenzugriff durch das Betriebssystem,
- Abschluss eines geschlossenen Programmteils,
- auftretende technische Fehler („exception“),
- längere Pause bei der Adressierung der Programmseite,
- I/Os,
- fehlende Daten,
- Status „idle“ mit timeout.

Es können aber auch Performance-Probleme in Erscheinung treten bei niedriger CPU-Auslastung. Dafür können folgende Gründe verantwortlich sein:

- Ein-/Ausgabekonflikte,
- hoher Zugriff auf Speichermedien durch gleichzeitig viele User (zentrale Dateien),
- zu hohe Paging-Rate,
- Engpässe durch Server-Tasks,
- ineffiziente Parallelverarbeitung durch zu wenig Tasks,

- CPU-Zeit für das Ausführen bestimmter Programme,
- zu viele System-Aufrufe,
- System-Selbstverwaltung,
- zu viele Bibliotheksroutinen,
- zu viele I/O-Routinen,
- zu viele gleichzeitige User,
- Multiprozessor-Management,
- ungünstige CPU Taktzahl pro Prozess,
- Prozessor-Stillstand,
- Abfolge von Maschinenbefehlen.

Falls Performance-Probleme weiter bestehen, selbst wenn die CPU „idle“ ist, so sind diese Probleme woanders zu suchen. Durch Aufrüstung der CPU ist dann nichts gewonnen.

### 2.3.1.1.3 Hauptspeicher

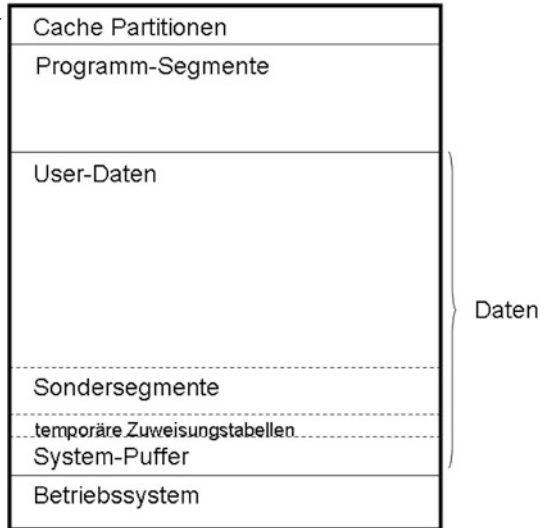
Ein wesentliches Merkmal für den Hauptspeicher ist seine nominelle Größe: die Anzahl von Zeichen, Bytes, die für eine zentrale Verarbeitung und die dazu erforderliche temporäre Speicherung zur Verfügung steht. Die absolute Größe des Hauptspeichers kann für eine Performance-Situation

- passend,
- zu groß oder
- zu klein.

sein. Das liest sich banal. Dabei gibt es Folgendes zu bedenken: ein überdimensionierter Hauptspeicher, häufig als Empfehlung bei Engpässen ausgesprochen, kann zu sekundären CPU-Problemen führen, wenn die Leistung nicht Schritt hält, oder eine Unterdimensionierung von I/O-Kanälen aufweisen. Der Grund liegt darin, dass die Anzahl möglicher Tasks, die geladen werden können, die Möglichkeiten dieser beiden anderen Ressourcen übersteigt, sodass Warteschlangen sich herausbilden.

Natürlich ist ein zu klein dimensionierter Hauptspeicher eher das Problem. Der Hauptspeicher hat umzugehen mit (Abb. 2.9):

**Abb. 2.9** Hauptspeicherbelegung



- Programm-Segmenten,
- Daten,
- Cache Partitionen und
- Teilen vom Betriebssystem.

Normalerweise ist ein Teil des Hauptspeichers ständig durch Grundfunktionen des Betriebssystems belegt: base memory mit einer fixen Prozentzahl des gesamten Speichers, z. B. 10 %. Weitere Anteile des Betriebssystems werden später je nach Bedarf nachgeladen. Ein gewichtiger Teil wird für die User-Prozesse benötigt. Deshalb ist z. B. bei Betriebssystem-Updates, die zusätzliche Funktionalitäten beinhalten können, darauf zu achten, dass entsprechend neu und sparsam konfiguriert wird.

Cache Partitionen werden benötigt für Speichersegmente, die Daten enthalten, auf denen wiederholt zugegriffen wird. Diese Partitionen sind konfigurierbar, können aber auch dynamisch zugewiesen werden. Im letzteren Fall stellt die Cache kein besonderes Problem dar, obwohl bei Engpässen Einschränkungen bezüglich des Cache-Anteils entste-

hen können, sodass sich die Performance wiederum verschlechtert, dadurch dass die Daten jetzt wieder direkt von den Platten gelesen werden müssen. Auf der anderen Seite führt eine intensive Cache-Nutzung zu Interrupt und Adressen Mapping. Dadurch entstehen zusätzliche CPU-Overheads.

Die Datenbelegung des Hauptspeichers speist sich aus folgenden Quellen:

- Userdaten,
- temporäre Zuweisungstabellen,
- Sondersegmente,
- System-Puffer.

Die Userdaten beaufschlagen den einem spezifischen User zugewiesenen Adressraum, zum Beispiel für Sortiervorgänge. Sondersegmente werden beispielsweise für Nachrichtendateien oder Kommunikationspuffer benötigt und System-Puffer für bestimmte System-Tasks. Der Rest steht dann Programmsegmenten oder für das Paging zur Verfügung.

Ein wichtiger Faktor bei der Betrachtung von Hauptspeicher-Performance ist die Paging-Rate. Mit Paging-Rate ist die Anzahl von Vorgängen pro Zeiteinheit gemeint, die aktuell Speicher residente Code-Seiten (Pages), die momentan für die Ausführung eines Programms durch die CPU benötigt werden, temporär aus dem Speicher entfernen, durch andere ersetzen, und die ursprünglichen Seiten später wieder hinein laden. Dieser Vorgang wird auch als Swapping bezeichnet.

Es gibt einige Algorithmen, die diejenigen Hauptspeichereinträge identifizieren, die dem Swapping unterliegen sollen. Normalerweise werden die Ressource-Warteschlangen bei jedem System-Zyklus innerhalb der Frequenz der System-Uhr abgefragt, um festzustellen, auf welche Segmente während des vorhergehenden Zyklus nicht zugegriffen wurde. Diese werden dann mit einem Flag als Overlay-Kandidaten versehen. Beim nächsten Zyklus werden die Overlay-Kandidaten nochmals geprüft und dann unter Umständen gewappt, falls Speicherplatz benötigt wird. Gibt es viele Overlay-Kandidaten und lange Verweilzeiten für bestimmte Segmente, die ohne Prozess-Unterbrechung gewappt werden können, entstehen nur minimale Memory Management Overheads.

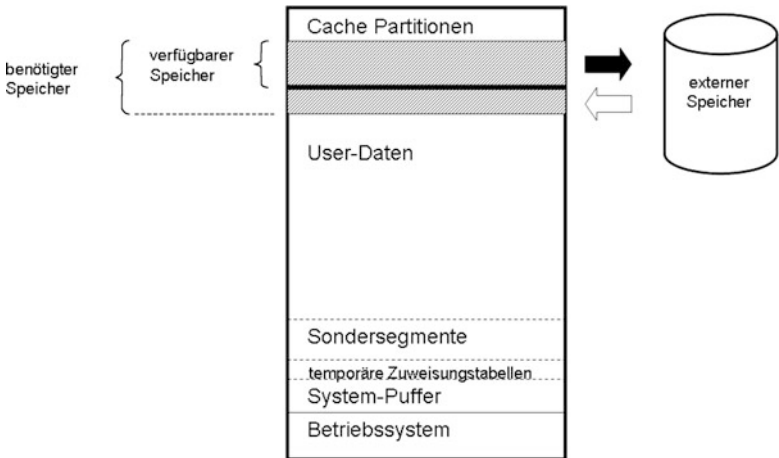


**Abb. 2.10** Paging

Hohe Paging-Raten können kritisch für eine effiziente Nutzung der Ressource Hauptspeicher werden (Abb. 2.10). Gründe dafür können sein:

- zu viele Prozesse,
- zu große Programme.

Deshalb geben Hersteller auch maximal empfohlene Werte an. Memory Management für eine sauber laufende CPU sollte nicht mehr als wenige Prozent betragen. Ansonsten führen zu hohe Paging-Raten grundsätzlich zu schlechtem Antwortzeitverhalten, weil CPU-Ressourcen dafür benötigt werden. Hintergrund: in den beiden genannten Fällen versucht die CPU zunächst, die Hauptspeicherbelegung neu zu organisieren, indem ein zusammenhängender Adressraum gebildet wird (das führt zu erheblichen Overheads). Falls das nicht gelingt, gibt die CPU auf, d.h. es kommt zu Unterbrechungen von Prozessen mit niedriger Priorität. Geswappt wird in den virtuellen Speicher oder auf Pagingbereiche der Festplatten. Konsequenzen:

- entgangene CPU Ressourcen für Memory Management (Erhöhung der subjektiven Antwortzeiten),

- unterbrochene Userprozesse (Erhöhung der subjektiven Antwortzeiten).

Wird dieses Problem chronisch, geht nichts um einen Upgrade des Hauptspeichers herum. Ein Grund dafür ist auch, dass externe Swapping-bereiche gesondert konfiguriert werden müssen. Sind diese zu klein, wird Paging zu einem Sekundärengpass mit zusätzlichen I/O-Wartezeiten und entsprechendem Memory Management.

Um Engpässe beim Hauptspeicher zu vermeiden, sind prophylaktisch regelmäßige Messungen mit einem geeigneten Monitor sinnvoll. Dabei sollten folgende Werte ermittelt werden:

- Anzahl der seitenwechselbaren Seiten (Größe des Hauptspeichers minus der Anzahl ständig für System-Zwecke residenten Seiten),
- Anzahl der global genutzten Seiten (Anzahl der seitenwechselbaren Seiten minus der Anzahl der frei nutzbaren Seiten).

Ähnlich wie bei der CPU gibt es auch für den Hauptspeicher Richtwerte. Auslastung bis 75 % gilt als unkritisch. Werden ständig über 90 % erreicht, ist ein Upgrade unerlässlich. Analog zur CPU gibt es ein Task-Management mit ganz ähnlichen Triggern über die Kontrollfunktionen der Hauptspeicherverwaltung mit Seitenverwaltungs-Algorithmus:

- Aktivierung,
- Deaktivierung,
- Zwangsdeaktivierung,
- Verdrängung.

Zusammenfassend können folgende Situationen entstehen:

Der Speicherbedarf ist höher als der vorhandene Speicher. Dann entsteht ein Paging-Problem. Das hat zur Folge, dass bestimmte Prozess-Schritte temporär auf dezentrale Speichermedien verlagert werden, was zu einem spürbaren Leistungsabfall führt. Lösungen können entweder durch Hardware-Erweiterung oder Anwendungsoptimierung herbeigeführt werden. Organisatorisch könnte man auch die Anzahl User reduzieren, was in der Regel nicht akzeptiert wird.

Das gegensätzliche Szenario ist, wenn ausreichend Speicherplatz vorhanden ist. Zwischen diesen beiden Zuständen gibt es keine Alternativen.

Dabei muss ein Programm nicht immer vollständig geladen sein, damit bestimmte Funktionen ausgeführt werden können. Programmseiten werden über deren Adressenfolgen stückweise geladen, was wiederum zu Swap-ins und Swap-outs, also Paging-Vorgängen führt. Ohnehin zeigt die Statistik, dass die meisten Programme 80 % ihrer Exekutionszeit in nur 20 % ihres eigenen Codes verbringen. Inaktive Prozesse werden hauptsächlich durch Online-Anwendungen hervorgerufen – wenn User vor dem Bildschirm sitzen und keine Eingaben tätigen, das Programm aber über die GUI dennoch aufgerufen bleibt. Zur Optimierung bezogen auf das Paging können folgende Maßnahmen herangezogen werden:

- Nutzung von gemeinsamen Subroutinen durch mehrere Programme,
- Größenoptimierung von Codes,
- Nutzung von gemeinsamen Bibliotheken,
- Speicherkonfigurierung anhand geschätzter User-Zahl.

## **2.3.2 Anwendungs-Performance**

### **2.3.2.1 Anwendungsparameter**

Obwohl ein Performance-Tuning auf der Basis von Anwendungsoptimierungen meistens zeitaufwendig ist und deshalb in der Regel kurzfristig keine sichtbaren Ergebnisse zeitigt, ist es dennoch häufig unerlässlich, um mittel- und langfristige Erfolge zu erzielen. Hier sind die Einflussfaktoren:

- Programmiersprache,
- modularer Programmaufbau,
- Anzahl Subroutinen und externer Aufrufe,
- Dateimanagement,
- Ein-/AusgabeprozEDUREN,
- GUI oder Batchverarbeitung,
- Größe von Codesegmenten,
- Kommunikations-Prozesse mit anderen Anwendungen.

Falls beeinflussbar, spielt die Programmiersprache eine wichtige Rolle für die zukünftige Performance einer Anwendung. Demgegenüber

sind zum Beispiel die Realisierungszeiten abzuwägen – modularer Programmaufbau – im Gegensatz zu struktureller Programmierung, die im Zuge moderner Sprachen an Bedeutung verloren hat. Gemeint ist in diesem Zusammenhang die Entkopplung von Anwendungsmodulen untereinander unter einem gemeinsamen Frontend. Dadurch wird der Hauptspeicher entlastet. Außerdem können dabei bestimmte Module aus unterschiedlichen Anwendungsanteilen angesprochen und so gemeinsam genutzt werden – zum Beispiel Fehlerroutrinen.

Während man erwarten kann, dass jedes neue Release akzeptable Anwendungs-Performance liefert, hängt das Ergebnis doch sehr stark ab von den vorbereitenden Maßnahmen. Dabei sollten die Wünsche der User berücksichtigt werden und deren Vorstellungen über gute Performance. Auch der gesamte Prozess-Ablauf, in dem die IT-Unterstützung eingebunden ist, sollte Berücksichtigung finden. Am Ende sollten folgende Fragen beantwortet werden:

- Muss die Anwendungsarchitektur angepasst werden, um bestimmten Performance-Kriterien zu genügen?
- Ist ein Upgrade der IT-Infrastruktur erforderlich?
- Wird Performance negativ beeinflusst durch GUI-Gestaltung?
- Wie wird sich eine zu erwartende Durchsatzsteigerung auf die System-Last auswirken?

Bestimmte Faktoren lassen sich manchmal mit geringem Aufwand tunen, wie z. B. Ein-/Ausgabeprozeduren oder Wechsel von GUI auf Batch. Andere erfordern größere Investitionen. Dazu gehören sicherlich die Transposition in andere Programmiersprachen oder Änderungen am Datenmodell.

Zusätzlich zu den Messwerten, die für die reine System-Performance von Bedeutung sind, gibt es eine Reihe von Kenngrößen, die im direkten Bezug zu den Anwendungen stehen. Dazu gehören insbesondere die Task spezifischen Betriebsmittelanforderungen im zeitlichen Durchschnitt und auf Spitzen, bezogen auf

- CPU-Zeit,
- Ein-/Ausgaben inklusive Zugriffe auf bestimmte Datentabellen und Plattenzugriffsstatistiken,

- Wartezeiten,
- Pagingraten,
- Kanal-Transporte inklusive TCP/IP-Verbindungen,
- eventuelle Kommunikationsereignisse im Rechnerverbund,
- Zugriffe auf Hauptspeicher,
- Antwortzeitenstatistik,
- Cache-Trefferquoten,
- Anzahl job-Aufrufe.

Diese Informationen werden entweder durch Messmonitore geliefert oder sind teilweise aus Logfiles zu ermitteln, wenn die entsprechenden Transaktionstypen dafür eingeschaltet sind. Logdateien für die Aufzeichnung von Transaktionen bedürfen in zweierlei Hinsicht spezifischer Ausrichtung:

- Formatierung,
- Selektion.

Bei der Formatierung ist darauf zu achten, dass die Einträge möglichst von technischen Informationen befreit werden. Das heißt alles, was Bezug zum Transaktionscode und zu Datenverschlüsselungen, Formatangaben usw. hat, sollte ausgespart werden. Lediglich aussagekräftige Textinformationen sollten hinterlegt werden, die dem Auswerter sofort einen Hinweis auf das tatsächliche Transaktionsgeschehen geben.

Die Selektion zur Aufzeichnung von Transaktionen sollte sich auf solche beschränken, die für eine spätere Analyse interessant sind. Nebensächliche Funktionsaufrufe wie „Datumsanzeige“, „Rückkehr zum Hauptmenu“ u. a. sollten ausgeblendet werden.

Neben Durchschnitt und Spitzen ist die Aufnahme des zeitlichen Verlaufs über typische Arbeitstage von Interesse. Hier zeigen sich zeitlich abhängige Engpässe, die unter Umständen zunächst einmal ohne großen technischen Aufwand organisatorisch entkoppelt werden können. Bei der Anzahl User, die hinter all diesen Ressourcenverbräuchen steht, ist zu differenzieren zwischen aktiven und lediglich über Sessions verbundenen.

## Weiterführende Literatur

1. Osterhage, W. 2012. *Performance-Optimierung*. Heidelberg: Springer Vieweg.

Mathematische Algorithmen und  
Computer-Performance kompakt

Osterhage, W.

2016, IX, 82 S. 28 Abb., Softcover

ISBN: 978-3-662-47447-1