Fachhochschule Kufstein Tirol

Studiengang Wirtschaftsinformatik

Leistungsanalyse von ERP-Systemen mit Fokus auf das ERP-System Semiramis

Entwicklung einer Vorgehensweise zur Leistungsanalyse im laufenden Betrieb

Diplomarbeit

Zur Erlangung des

Akademischen Grades Magister (FH)

Eingereicht von:	Günther Lanner
	Geburtsort: Wörgl
Erstgutachter:	Dr. Johannes Lüthi
Zweitgutachter:	Dr. Martin Delp
Ort, Datum:	Kufstein, den 30. Juni 2006

Eidesstattliche Erklärung

„Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Diplomarbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.“

_____

Günther Lanner

Inhaltsverzeichnis

Abkürzungsverzeichnis

AG	Aktiengesellschaft
AIX	Advanced Interactive eXecutive
API	Application Programming Interface
AS	Application Server
AWE	Address Windowing Extension
BIS	Business Integration Service
CBO	Cost Based Optimizer
CD	Compact Disk
CIS, C.I.S.	Cross Industrie Software
CISC	Complex Instruction Set Computing
CKPT	Checkpoint
CPU	Central Processing Unit
CQE	Classic Query Engine
DB2 UDB	Database 2 Universal Database
DBMS	Datenbank-Managementsystem
DBWR	Database Writer
DMA	Direct Memory Access
DTA	Database Tuning Advisor
E/A	Ein-/Ausgabe
ECC	Error Correcting Code
ERP	Enterprise Resource Planning
EVI	Encoded Vector Indizes
FH	Fachhochschule
Fibu	Finanzbuchführung
FIFO	First in First Out
GB	Gigabyte
GBit	Gigabit
GC	Garbage Collection
GUID	Global Unique Identifier

HB	Higher is Better
HTTPS	Hyper Text Transfer Protocol Secure
I/O	Input/Output
IA32, bzw. IA64	Intel Architecture 32, bzw. Intel Architecture 64
IBM	Industrial Business Machines
ICMP	Internet Control Message Protocol
IDE	Integrated Drive Electronics
IOP	Input Output Processor
IRQ	Interrupt Request
ISA	Industry Standard Architecture
ISP	Internet Service Provider
IT	Informationstechnologie
JDBC	Java Database Connectivity
JIT	Just in Time
JMX	Java Management Extension
JVM	Java Virtual Machine
kB	kiloByte
KTW	Karner, Thuile, Wolf
LAN	Local Area Network
LB	Lower is Better
LDM	Logical Disk Manager
LGWR	Log Writer
LRU	Least Recently Used
MB	Megabyte
Mbit	Megabit
MMU	Memory Management Unit
MRU	Most Recently Used
MS	Microsoft
ms	Millisekunden
MTU	Maximum Transfer Unit
NB	Nominal is Best

ns	Nanosekunden
ODBC	Open Database Connectivity
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
OQL	Object Query Language
PC	Personal Computer
PCB	Process Control Block
PCI	Peripheral Component Interconnect
PID	Process Identification
PMON	Process Monitor
RAID	Redundant Array of Inexpensive Disks
RISC	Reduced Instruction Set Computing
s	Sekunden
SAP	Systeme, Anwendungen und Produkte in der Datenverarbeitung
SAS	Semiramis Application Server
SGA	System Global Area
SLIC	System Licensed Internal Code
SMON	System Monitor
SNMP	Simple Network Management Protocol
SOM	Semiramis Output Manager
SQE	SQL Query Engine
SQL	Structured Query Language
SSL	Secure Socket Layer
TB	Terabyte
TIMI	Technology Independent Machine Interfaces
VPN	Virtuelles Privates Netzwerk
XML	Extensible Markup Language

Abbildungsverzeichnis

1 EINFÜHRUNG

1.1 Problemstellung

Enterprise Resource Planning (ERP)-Systeme sind in der Regel verteilte Systeme. Darunter versteht man die Verteilung der Komponenten eines ERP-Systems auf unterschiedliche Rechner im Netzwerk auf der einen, und auf unterschiedliche Architekturschichten (Präsentations-, Applikations- und Datenbankschicht, siehe dazu Abschnitt 2.2.3.1) auf der anderen Seite. Für die Leistungsanalyse ist also kein zentraler Einstiegspunkt vorhanden, vielmehr müssen die einzelnen Subsysteme mit individuellen Leistungsmonitoren analysiert werden. Die meisten ERP-Systeme bieten zwar die Möglichkeit einer zentralen Leistungsanalyse, jedoch reicht diese oft nicht aus, da detaillierte Analysen auf Betriebssystem- und Datenbankseite nicht möglich sind. Dieses zentrale Leistungsmonitoring ist je nach ERP-System Hersteller mehr oder weniger umfangreich. So ist beispielsweise im ERP-System SAP ein detaillierter Leistungsmonitor implementiert, der es erlaubt Leistungsdaten von Betriebssystem, Datenbank und SAP-Basis zu analysieren (Schneider, 2005). Aber auch hier muss zur detaillierten Analyse auf die Leistungsmonitore der einzelnen Subsysteme zurückgegriffen werden.

Für das ERP-System Semiramis, das im Folgenden als Fallstudie für die Leistungsanalyse eines ERP-Systems dient, stehen folgende Leistungsmonitore zur Verfügung:

- Der zentrale Leistungsmonitor im ERP-System: Dieser dient vor allem zur Analyse der Datenbank und zur Identifizierung von Anwendungen mit hoher Ausführungszeit. Aber auch Leistungsdaten des Semiramis Applikation-Servers (siehe Abschnitt 2.2.6) finden sich hier.
- Ein Datei-Leistungsmonitoring, welches detailliertere Analysen wie der zentrale Leistungsmonitor zu Antwortzeiten von Anwendungen mit hoher Ausführungszeit bietet.
- *OpenNMS*: Ein im Rahmen einer Diplomarbeit entwickeltes Java Programm, welches über das Simple Network Management Protokoll (SNMP) leistungsrelevante Daten aus Semiramis, der Java Virtual Machine, aber auch aus Betriebssystemen und Datenbanken ausliest (Wurzrainer, 2006).
- Leistungsmonitore der einzelnen Subsysteme: Es sind dies Leistungsmonitore für Betriebssysteme, Datenbanken, der Java Virtual Machine (JVM) und für Netzwerke. Semiramis ist vollständig mit Java entwickelt worden, und kann auf verschiedenen Plattformen installiert werden. Als Betriebssysteme kommen Microsoft Windows Serverprodukte, Linux, sowie i5/OS von IBM; als Datenbanken Oracle, DB2 UDB for iSeries sowie Microsoft SQL-Server in Frage.

Diese oben genannten Werkzeuge zur Leistungsanalyse liefern im Einzelnen zuverlässige Ergebnisse, jedoch ein vollständiges zentrales Werkzeug bzw. eine generelle plattformübergreifende Vorgehensweise zur Leistungsanalyse ist nicht vorhanden.

1.2 Zielsetzung und Motivation

Das Ziel dieser Arbeit ist es, eine Vorgehensweise für die Leistungsanalyse des ERP-Systems Semiramis mit den vorhandenen Monitoring-Möglichkeiten zu entwickeln. Tätigkeiten, die früher unstrukturiert – nach dem Ermessen und Kenntnisstand des jeweiligen Technikers – durchgeführt wurden, sollen nach einer allgemeinen Vorgehensweise durchgeführt werden können. Somit soll sich eine wesentliche Zeitersparnis bei der Analyse von Problemen ergeben. Auch in Hinsicht auf die Vielzahl der verwendeten Plattformen, und der derzeitigen Spezialisierung von Mitarbeitern auf einzelne Plattformen kann sich ein großer Nutzen ergeben, da Analysen auch von Mitarbeitern mit Schwerpunkten auf anderen Plattformen durchgeführt werden können. Diese Vorgehensweise für die Leistungsanalyse des ERP-Systems Semiramis soll auch auf andere ERP-Systeme angewendet werden können. Des Weiteren wird ein Grobkonzept zur Integration weiterer Leistungsdaten in Semiramis erstellt. Dieses Grobkonzept wird ebenfalls auf die Anwendbarkeit auf andere ERP-Systeme geprüft.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in drei Teile: die theoretischen Grundlagen, die Verbindung Theorie – Praxis, sowie die Lösungsvorschläge.

Im ersten Teil werden die wissenschaftlichen Grundlagen für den weiteren Verlauf erarbeitet. Es werden Begriffe definiert und für Leistungsaspekte relevante Architekturen von Betriebssystemen, Datenbanken und Netzwerken beschrieben. Dabei werden die Besonderheiten der einzelnen Betriebssystem- und Datenbankprodukte vorgestellt. Auch werden die Java Virtual Machine und die Semiramis-Architektur beschrieben. Anschließend werden Performancekennzahlen definiert, Monitore klassifiziert, sowie ein systematischer Ansatz einer Performanceanalyse vorgestellt. Der zweite Teil dieses Kapitels orientiert sich an (Jain, 1991) und wird durch aktuelle Literatur ergänzt.

Im zweiten Teil folgt die Vorgangsweise zur Leistungsanalyse eines Semiramis-Systems. Zu Beginn werden die in Semiramis implementierten Möglichkeiten vorgestellt. Es folgen Vorgangsweisen zur Analyse von Hardware, Betriebssystem, Datenbanken, Netzwerken und der Java Virtual Machine. Es werden spezifische Probleme vorgestellt und gezeigt, mit welchen Möglichkeiten diese für die einzelnen Plattformen identifiziert werden können. Dieser Teil orientiert sich an Methoden zur Analyse von SAP-Systemen (Schneider, 2005; Anderson, 2005a) und Empfehlungen aus der Semiramis Dokumentation (CIS, 2004; CIS, 2004b; CIS, 2005a; CIS, 2005b; CIS, 2006; CIS, 2006a; CIS, 2006b; CIS, 2006c; CIS, 2006d). Für jede Betriebssystem- bzw. Datenbankplattform findet sich eine Aufstellung wichtiger Performancekennzahlen und Monitoring-Tools. Am Ende jedes Abschnittes finden sich Vorgehensweisen zur Leistungsanalyse anhand von Flussdiagrammen. Für die Analyse der Hardware finden sich Gegenüberstellungen von Performancekennzahlen, Monitoring-Tools, Grenzwerten und Handlungsempfehlungen für die unterschiedlichen Betriebssysteme.

Aus den Erkenntnissen des zweiten Teiles, werden im dritten Teil Lösungsvorschläge präsentiert. Vier Lösungsvorschläge werden vorgestellt, wobei zwei Vorschläge detaillierter beschrieben

werden. Es sind dies die Erstellung einer Dokumentvorlage und ein Grobkonzept zur Integration weiterer Leistungsdaten in Semiramis. Diese zwei Vorschläge werden auch auf ihre Anwendbarkeit auf andere ERP-Systeme überprüft.

2 THEORETISCHE GRUNDLAGEN

Im diesem Kapitel werden wissenschaftliche Grundlagen erarbeitet. Zunächst sei der Begriff Leistung – in der Arbeit wird auch immer wieder das englische Synonym Performance vorkommen – definiert, wie er in dieser Arbeit verstanden wird. Das folgende Zitat stellt keine Definition im allgemeinen Sinne dar, beschreibt aber den Begriff Performance für diese Arbeit sehr treffend:

> *„Die Performance eines Datenverarbeitungssystems ist als die Fähigkeit definiert, gegebene Anforderungen an Antwortzeiten und Datendurchsatz zu erfüllen. Solche Anforderungen können z.B. sein, dass innerhalb einer Stunde ein Durchsatz von 10000 gedruckten Rechnungen erreicht werden muss oder dass die Antwortzeit für das Erfassen eines Kundenauftrages unter einer Sekunde liegen soll. Eine gute Performance ist keine absolute Eigenschaft einer E-Business Anwendung, sondern immer relativ zu den Anforderungen an diese zu sehen“ (Schneider, 2005, S. 17).*

2.1 Allgemeine Begriffe und Abgrenzungen

2.1.1 ERP-Systeme

ERP-Systeme ermöglichen Unternehmen eine integrative Abwicklung ihrer Geschäftsprozesse und eröffnen dadurch umfassende Möglichkeiten zur Verbesserung ihrer Abläufe. Die Daten der verschiedenen betriebswirtschaftlichen Bereiche werden zentral in einer Datenbank gespeichert. Dadurch ist eine bereichsübergreifende Nutzung der Daten möglich, ohne dass sie mehrfach eingegeben und gepflegt werden müssen. Beispielsweise wird bei der Buchung eines Wareneingangs dieser Vorgang automatisch auch in den Konten der Finanzbuchhaltung (Fibu) wertmäßig erfasst. Die Funktionsbereiche eines ERP-Systems beinhalten Vertrieb, Personalwirtschaft, Produktion, Materialwirtschaft, Logistik, Finanz- und Rechnungswesen, Controlling, sowie Querschnittsanwendungen wie Workflow, Archivierung und Reporting (Schwarzer, 2004, S. 131f).

Von der Gartner Group wurde der Begriff ERP II geprägt, unter den sich Semiramis einordnet. Unter ERP II versteht man die web-zentrierte, technisch integrierte Zusammenarbeit (Collaboration) auf der Basis des Internets. ERP II Systeme ermöglichen somit unternehmensübergreifende Prozesse wie z.B. Supply Chain Management einfacher abzubilden (Gartner, 2000).

Semiramis unterscheidet sich von den obigen Definitionen dahingehend, dass für Finanz- und Rechnungswesen, Controlling, Reporting, sowie für die Personalwirtschaft Produkte von Drittanbietern verwendet werden, die über Schnittstellen an das System angebunden werden. Diese

Drittanbieter-Produkte, mit Ausnahme des Reportings, sind nicht Gegenstand dieser Arbeit. Eine detaillierte Beschreibung der Semiramis-Architektur findet sich in Abschnitt 2.2.6.

2.1.2 Stamm-, Bewegungs- und Customizingdaten

Unter Stammdaten versteht man all jene Datenbestände, die nur im Ausnahmefall verändert werden, wie z.B. Material. Zu Material zählen Roh-, Hilfs- und Betriebsstoffe sowie Halb- und Fertigfabrikate (Mertens, 2001).

Bewegungsdaten lassen sich in Vormerkdaten, Transferdaten und Archivdaten unterteilen (Mertens, 2001):

- Vormerkdaten (offene Posten): Diese Daten werden nur vorübergehend benötigt. Es existiert ein geplanter Zeitpunkt, bei dessen Eintreffen sie gelöscht werden. Beispiele sind eröffnete Angebote oder Beschaffungsaufträge.
- Transferdaten: Daten, die von einem Programm generiert werden und dann einem anderen Programm geliefert werden. Ein konkretes Beispiel für Semiramis sind Finanzbuchführungs-Daten (Fibu-Daten), die über eine Schnittstelle der Fibu-Software Varial zur Verfügung gestellt werden.
- Archivdaten: Dazu zählen abgespeicherte Vergangenheitswerte, insbesondere Zeitreihen und historische Aktionen. Ein Beispiel sind Auftragseingänge eines Produktes in den letzten 36 Monaten. Solche Daten sind vorzugsweise in Data Warehouse-Systemen anzutreffen, werden in dieser Arbeit also nicht behandelt.

Eine beispielhafte Beschreibung von Stamm-, Bewegungs- und Customizingdaten findet sich in (Schneider, 2005), die sich in der Definition der Bewegungsdaten von der obigen Definition unterscheidet:

Stammdaten sind z.B. Artikel, Kunden oder Lieferanten. Sie wachsen in der Regel langsam, können aber im Laufe der Zeit sehr umfangreich werden. Größtenteils wird lesend auf Stammdaten zugegriffen.

Bewegungsdaten sind z.B. Vertriebsaufträge oder Beschaffungsaufträge. Diese Daten wachsen linear mit der Zeit an, es wird überwiegend schreibend auf diese Daten zugegriffen.

Customizingdaten bilden unter anderem die Geschäftsprozesse des Unternehmens ab. Zum Customizing gehört z.B. die Definition der Mandanten oder der Nummernkreise. Customizingdaten sind wenig umfangreich, und werden selten geändert. Es wird also in der Regel nur lesend auf sie zugegriffen.

Aufgrund der oben genannten verschiedenen typischen Zugriffsarten sind zur Verwaltung von Stamm-, Bewegungs- und Customizingdaten unterschiedliche Strategien erforderlich, um Performance zu optimieren. Dies wird einen Schwerpunkt der Arbeit bilden.

2.2 Architekturen

Im Folgenden werden die grundlegenden Architekturen von Betriebssystemen, Netzwerken und Datenbanken hinsichtlich leistungsrelevanter Aspekte erläutert. Performanceprobleme können an unzähligen Stellen eines Systems auftreten, die Ursachen sind oft schwer zu finden. So umfangreich das Thema auch ist, lässt sich dennoch eine grundlegende Struktur erkennen. Ein Performanceproblem lässt sich oft auf eine suboptimale Nutzung einer Ressource zurückführen. So unterscheiden sich die Geschwindigkeiten von Central Processing Units (CPU's), Caches, Bussystemen, Netzwerken, Primär- und Sekundärspeichern um wesentliche Faktoren. Ein Zugriff auf eine Festplatte benötigt eine ca. eine Million Mal längere Zeitspanne wie ein Zugriff auf einen Hauptspeicher. Die Festplatte hat jedoch eine wesentlich größere Kapazität. Da ERP-Systeme aufgrund ihrer zugrunde liegenden Datenbanken dazu tendieren viele Ein-/Ausgabe (E/A oder I/O)-Zugriffe zu verursachen und verhältnismäßig weniger CPU Leistung zu beanspruchen, fällt dies noch mehr ins Gewicht. Es muss also immer ein Weg gefunden werden, bestmögliche Leistung zu erzielen, aber trotzdem mit den begrenzten Speicherkapazitäten der einzelnen Ressourcen auszukommen. Dies ist auch hinsichtlich monetärer Aspekte zu bedenken, da Hauptspeicher ein Vielfaches einer Festplatte kostet. In Abbildung 1 findet sich eine Aufstellung typischer Zugriffszeiten und Kapazitäten. Die Werte sind grobe Schätzungen.

Abbildung 1 – Eine typische Speicherhierarchie (modifiziert nach Tanenbaum, 2002, S. 36).

Der folgende Abschnitt beschäftigt sich mit Strategien, diese Ressourcen bestmöglich zu verwalten.

2.2.1 Betriebssysteme

Ein Betriebssystem hat drei Funktionen: Zum einen die Erweiterung der Hardware, d.h. dem Benutzer eine virtuelle Maschine zur Verfügung zu stellen, die den Zugriff auf die Hardware vereinfacht. Zum anderen die Verwaltung von Ressourcen, d.h. eine geordnete Zuteilung und Kontrolle der Prozessoren, Speicher sowie der Ein-/Ausgabegeräte. Diese zweite Aufgabe eines Betriebssystems wird im Folgenden näher betrachtet, es sind dies die Speicherverwaltung, Prozessverwaltung und die Ein-/Ausgabeverwaltung. Auf die dritte Aufgabe, die Dateiverwaltung, wird hier nicht näher eingegangen. Dieser Abschnitt orientiert sich an (Tanenbaum, 2002). Weiterführende Quellen werden explizit erwähnt.

2.2.1.1 Prozessverwaltung

CPU

Die CPU ist das „Gehirn“ des Computers. Sie holt sich Befehle aus dem Speicher und führt sie aus. Jede CPU besitzt eine Menge von Anweisungen, die sie ausführen kann. Deshalb kann z.B. ein Complex Instruction Set Computing (CISC) Prozessor – wie der Pentium von Intel – keine Programme von einem Reduced Instruction Set Computing (RISC) Prozessor – wie dem Power 5 von IBM – ausführen. Weil der Speicherzugriff zum Holen eines Befehls oder Daten sehr viel länger dauert als die Ausführung der Anweisung, besitzen alle Prozessoren interne Register und Caches (Level1, Level2 und optional einen Level3 Cache), damit wichtige Variablen und Ergebnisse innerhalb der CPU gespeichert werden können. Somit kann die Leistung von Prozessoren nicht direkt verglichen werden. Sie hängt von mehreren Faktoren ab, der Art des Prozessors (CISC oder RISC), der Taktfrequenz, der Menge von Level1, Level2 und Level3 Caches, und deren Nähe zum Prozessor.

Prozesse

Das Prozessmanagement verwaltet die Prozesse auf einem Computer. Es ist zuständig für das Erzeugen und Zerstören von Prozessen, verwaltet Prozesszustände, und das Scheduling von Prozessen. Ein Prozess wird über seinen Process Control Block (PCB) vom Betriebssystem identifiziert. Der PCB enthält unter Anderem eine Prozessidentifikation, den Prozesstyp (user, system, usw.), die Prozesspriorität, Statusinformationen und Ressourceanforderungen, wie Speicher, Festplatten und dergleichen. Multitasking-Betriebssysteme können scheinbar Prozesse gleichzeitig verarbeiten. Genau betrachtet läuft aber – wenn man ein Einprozessor System betrachtet – zu einem gewissen Zeitpunkt jeweils nur ein Prozess auf der CPU. Um zu bestimmen, welcher Prozess vom wartenden in den laufenden Zustand wechselt, werden Scheduling Algorithmen benötigt. Der Scheduler verwaltet Warteschlangen von Prozessen, die mit gewissen Prioritäten versehen sind (Fortier, 2003, S. 67ff). Hierzu gibt es verschiedenste Verfahren, auf die in Abschnitt 2.2.1.4 der einzelnen Betriebssysteme näher eingegangen wird.

Die Änderung der Prozessorzuteilung wird context switch oder Prozesswechsel genannt. Der aktive Prozess gibt den Prozessor entweder freiwillig frei, wird von einem Hardware Interrupt unterbrochen, oder bekommt ihn vom Scheduler entzogen. Dieser Vorgang wird vom Dispatcher gesteuert (Märtin, 2003, S. 180). Der Prozesswechsel ist ein ressourcenintensiver Vorgang, da die PCB beider beteiligter Prozesse gespeichert werden müssen. Die Performance eines Prozesses ist also nicht nur abhängig von der Leistungsfähigkeit der CPU, sondern auch vom jeweiligen Scheduling-Verfahren des Betriebssystems, sowie von der Geschwindigkeit und Größe des Hauptspeichers, um den PCB bei einem context switch speichern zu können. Siehe dazu Abschnitt 2.2.1.2.

Threads

Ein alternatives Konzept eines Prozesses, welches unter Anderem in Java und somit auch in Semiramis verwendet wird, ist sein „Ausführungsfaden“, üblicherweise als Thread bezeichnet. In einem Prozess können mehrere Threads parallel laufen. Sie teilen den Adressraum, geöffnete Dateien und andere Ressourcen mit dem Prozess. Dieses Konzept wird als Multithreading bezeichnet. Auch hier wechselt die CPU schnell zwischen den Threads, und erzeugt so eine scheinbare Parallelität. Laufen z.B. drei Threads in einem Prozess, und angenommen es ist nur ein Prozess aktiv am System, so steht einem Thread ein Drittel der CPU Zeit zur Verfügung. Threads bringen somit Performance-Vorteile, da sie weniger Ressourcen allokieren müssen. Ein Thread wird daher oft auch als „leichtgewichtiger Prozess“ bezeichnet.

Moderne Prozessoren – wie z.B. die Intel XEON Reihe, Intel bezeichnet diese als Hyperthreading-Prozessoren – unterstützen hardwareseitiges Multithreading, d.h. sie stellen zwei virtuelle Prozessoren innerhalb einer CPU zur Verfügung. Dies bringt weitere Vorteile in der Performance, da so beispielsweise ein Zwei-Prozessorsystem dem Betriebssystem als Vier-Prozessorsystem erscheint und somit context switches des Betriebssystems halbiert werden (Märtin, 2003, S. 30). Hardwareseitiges Multithreading muss aber vom Betriebssystem unterstützt werden, damit es genutzt werden kann.

2.2.1.2 Speicherverwaltung

Die Speicherverwaltung verwaltet die Speicherhierarchie des Computers. Es muss das Ein- und Auslagern von Daten in den Primärspeicher und auch die Verwaltung des freien Speicherplatzes organisiert werden. Um diese Funktionen ausführen zu können, wird der Speicher üblicherweise in Blöcke mit fixer Länge (Pages oder Seiten), oder in Blöcke mit variabler Länge (Segmente), aufgeteilt. Es muss Speicher für Prozesse bei ihrer Initialisierung allokiert, bzw. bei ihrer Beendigung deallokiert werden, und der Speicherplatz periodisch ausgeräumt werden wenn der Speicher fragmentiert wird (Fortier, 2003, S. 72f).

Da das Ein- und Auslagern mit hohem Zeitaufwand verbunden ist, müssen die Ein- und Auslagerungsvorgänge minimiert werden. Zur Speicherverwaltung gibt es zwei unterschiedliche Strategien, Swapping und virtueller Speicher. Die Swapping Strategie ist veraltet und wird nur noch von einigen UNIX Versionen verwendet, deshalb wird diese hier nicht näher beschrieben.

Virtueller Speicher mit Paging

Bei dieser Strategie können Programme auch dann laufen, wenn sich nur ein Teil von ihnen im Hauptspeicher befindet. Die Grundidee ist es, zu erlauben dass der Programmcode, die Daten und der Stack größer sind als der verfügbare Hauptspeicher. Das Betriebssystem hält die Teile des Programms, die gerade gebraucht werden im Hauptspeicher, der Rest wird auf den Sekundärspeicher ausgelagert.

Die meisten Systeme mit virtuellem Speicher verwenden die Technik des Paging. Die Adressierung geht nicht direkt an den Speicher, sondern an die Memory Management Unit (MMU), die virtuelle Adressen (Adressen, die vom Programm verwendet werden) auf die

physischen Adressen abbildet. Der virtuelle Adressraum ist in Einheiten unterteilt, die Seiten (Pages) genannt werden, die entsprechenden Einheiten im physischen Speicher heißen Seitenrahmen. Seiten und Seitenrahmen sind immer gleich groß, z.B. 4kB. Es werden immer nur ganze Seiten übertragen. Wenn ein Programm versucht, auf eine Seite zuzugreifen, die nicht im Speicher liegt, wird ein Systemaufruf gestartet, der Seitenfehler genannt wird. Dieser lagert einen wenig benutzen Seitenrahmen auf die Festplatte aus, und lagert die benötigte Seite ein. Um die Seite auszuwählen, die ausgelagert werden soll, gibt es unterschiedliche Strategien:

- First in First Out (FIFO): Die älteste Seite wird bei einem Seitenfehler ausgelagert.
- Least Recently Used (LRU): Bei einem Seitenfehler wird die am längsten unbenutzte Seite ausgelagert.
- Most Recently Used (MRU): Bei einem Seitenfehler wird die am letzten benutzte Seite ausgelagert. Diese Strategie mag auf den ersten Blick unsinnig erscheinen, hat aber bei Datenbanken durchaus Vorteile, siehe dazu Abschnitt 2.2.2.

2.2.1.3 Ein-/Ausgabeverwaltung

Hauptaufgabe der Ein-/Ausgabeverwaltung ist es, die Nutzung von Ein-/Ausgabegeräten eines Computers zu kontrollieren und zu überwachen. Außerdem sollte eine Schnittstelle zwischen den Geräten und dem Rest des Systems zur Verfügung stehen, die einfach zu bedienen ist. Besonderes Augenmerk der Arbeit liegt auf Festplatten, deshalb folgt zunächst ein kurzer Einblick in die Architektur von Plattenspeichern.

Plattenspeicher

Plattenspeicher werden in Zylinder aufgeteilt, wobei jeder Zylinder so viele Spuren besitzt, wie Köpfe vertikal angeordnet sind. Die Spuren sind wieder in Sektoren eingeteilt. Auf modernen Festplatten gibt es mehrere tausend Zylinder, 10-20 Köpfe und mehrere hundert Sektoren pro Spur. Jeder Sektor hat eine bestimmte Größe, z.B. 512 Bytes.

Abbildung 2 zeigt eine vereinfachte Darstellung, wie auf einen Sektor der Festplatte zugegriffen wird.

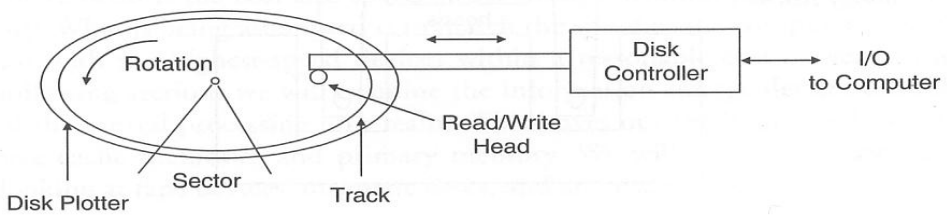


Abbildung 2 – Schematische Übersicht eines magnetischen Plattenspeichers (Quelle: Fortier, 2003, S. 52).

Angenommen, der Disk Controller weiß, wo sich der gewünschte Block befindet, dann muss sich der Plattenarm zum gewünschten Zylinder bewegen. Dies wird als Suchzeit (*t{seek}*) bezeichnet. Am Plattenarm befindet sich der Schreib-/Lesekopf. Wenn der Kopf am gewünschten Zylinder ist,

muss der Sektor gefunden werden. Dieser wird durch die Rotation der Platte am Kopf vorbeibewegt. Diese Zeit wird als Rotationszeit (*t{rotate}*) bezeichnet. Die durchschnittliche Rotationszeit kann mit der Formel (1/ Rotationsgeschwindigkeit*0,5) berechnet werden. Je höher die Rotationsgeschwindigkeit, desto kleiner ist diese Zeit. Wenn der Arm über dem gewünschten Sektor ist, können die Daten übertragen werden. Diese Zeit heißt Transferzeit (*t{transfer}*). Die Transferzeit ist abhängig von der Rotationsgeschwindigkeit und der gelesenen Datenmenge. Somit erhält man für die durchschnittliche Zeit, um Daten von einer Festplatte abzurufen: *T = t{seek} + t{rotate} + t{transfer}*. (Fortier, 2003, S. 52f). Die oben angeführten Daten sollten in den Spezifikationen einer Festplatte angegeben sein. Oft ist in den Spezifikationen auch die Datentransferrate einer Festplatte angegeben. Um *t{transfer}* zu ermitteln, ist die durchschnittliche Datengröße durch die Datentransferrate zu dividieren. Die durchschnittliche Datengröße ist applikationsabhängig und muss aus Erfahrungswerten oder durch Monitoring von Schreib- und Lesevorgänge der Festplatten über einen längeren Zeitraum ermittelt werden.

Da CPU und Hauptspeicher wesentlich schneller arbeiten als Festplatten, ist es sehr wahrscheinlich, dass es beim Schreiben und Lesen auf die Festplatten zu Wartezeiten kommen kann. Dazu verwaltet das Betriebssystem Warteschlangen für die Datenträger. Eine Möglichkeit, die Wartschlange für das Betriebssystem zu verkürzen, ist es, eine Festplatte zusätzlich mit einem schnellen Cache Speicher auszustatten. Wenn die Daten im Cache abgelegt sind, wird dem Betriebssystem signalisiert, dass die Daten bereits auf die Festplatte geschrieben sind. Es muss aber gewährleistet sein – durch die Verwendung einer Batterie –, dass dieser Cache auch nach einem Stromausfall wieder zur Verfügung steht.

Um einerseits Performance und andererseits die Ausfallssicherheit von Festplatten zu erhöhen, werden Redundant Array of Inexpensive Disks (RAID) Systeme verwendet (Silberschatz, 2005, S. 470ff). Bei RAID-Systemen werden Daten über mehrere Festplatten verteilt, was als striping bezeichnet wird. Es ist ein RAID Controller erforderlich, der hardware- oder softwaremäßig implementiert sein kann. Die gängigsten Level sind RAID0, RAID1, RAID5 und RAID10.

- RAID0: Es werden mehrere Festplatten zusammengeschlossen, und Schreib- bzw. Leseoperationen parallel ausgeführt. Es werden jeweils so genannte Strips geschrieben, ein Strip sind k Sektoren. Dieser Level arbeitet am Besten mit großen zusammenhängenden Daten. RAID0 bietet als einziger RAID-Level keine Redundanz, dafür aber erhöhte Lese- und Schreibgeschwindigkeiten, da mehrere Operationen parallel ausgeführt werden können.
- RAID1: Bei dieser Variante werden alle Platten verdoppelt. Bei einem Schreibvorgang wird jeder Strip doppelt geschrieben. Bei einem Lesevorgang können beide Kopien genutzt werden. Deshalb ist bei RAID1 das Lesen annähernd doppelt so schnell wie das Schreiben.
- RAID5: Die Nutzdaten werden auf alle Festplatten verteilt. Zusätzlich werden Paritätsinformationen (Fehlerkorrekturinformationen) zu den Nutzdaten abgespeichert. Diese Paritätsinformationen liegen auf jeweils anderen Platten als die Nutzdaten. Das Lesen von Daten ist hier wieder sehr schnell. Das Schreiben hingegen langsam, da zusätzlich zu den Daten Paritätsinformationen geschrieben werden müssen. RAID5 ist

dennoch der gängigste RAID-Level, da mit einer relativ niedrigen Anzahl an Festplatten, gute Ausfallssicherheit und Performance beim Lesen erreicht wird.

- RAID10: Dies ist eine Kombination aus RAID1 und RAID0, im Prinzip ein doppelt vorhandenes RAID1. Somit erhält man Vorteile beider Varianten, erhöhte Performance durch RAID0 und erhöhte Ausfallssicherheit durch RAID1, hat aber auch höhere Kosten durch die doppelte Anzahl an benötigten Festplatten.

Begriffe zur Ein-/Ausgabe

Um die unterschiedlichen Hardware Komponenten eines Computers untereinander zu verbinden, werden Bussysteme verwendet. Da sich die Geschwindigkeiten der einzelnen Komponenten stark unterscheiden, sind beispielsweise auf einem Personal Computer (PC) der Speicherbus, der Peripheral Component Interconnect (PCI) Bus für Grafikkarten, der Industry Standard Architecture (ISA) Bus für Drucker, Soundkarten und dergleichen, oder der Integrated Drive Electronics (IDE) Bus für Festplatten und CD-Laufwerke vorhanden.

Um mit den Ein-/Ausgabegeräten zu kommunizieren, besitzt jedes Gerät einen Controller. Beispielsweise ist der Controller einer Festplatte verantwortlich dafür, den seriellen Bitstrom einer Festplatte – bestehend aus einem Vorspann, den Daten des Sektors und einem Error Correcting Code (ECC) zur Fehlerkorrektur –, in Byte-Blöcke umzuwandeln, die vom Betriebssystem verarbeitet werden können. Auch Controller haben meistens Caches eingebaut.

Um dem Betriebssystem zu signalisieren, dass ein Gerät eine Aufgabe beendet hat, werden Interrupts verwendet. Ein Interrupt veranlasst die CPU, ihre aktuelle Aufgabe zu unterbrechen, und die Anforderung des Geräts zu bearbeiten. Stehen mehrere Interrupts an, so werden diese der Reihe nach bearbeitet, außer ein Interrupt mit höherer Priorität steht an. Es kann hier also zu Wartezeiten kommen, die mit Warteschlangen verwaltet werden.

Hardwaregeräte werden meistens nicht direkt adressiert, sondern über so genannte Memory Mapped Ein-/Ausgabe. Hier werden die Adressen aller Ein-/Ausgabegeräte in den Hauptspeicher eingeblendet. Um die CPU bei einem Hardwarezugriff zu entlasten, werden Direct Memory Access (DMA) Controller verwendet, die die Kommunikation mit den Ein-/Ausgabegeräten durchführen.

2.2.1.4 Besonderheiten der einzelnen Betriebssysteme

Windows 2003 Server

In diesem Abschnitt erfolgt ein kurzer Einblick in die Besonderheiten des Windows 2003 Server Betriebssystems. Weiterführende Hinweise finden sich in (Boswell, 2005). Windows 2003 Server wird in einer 32 Bit Version (IA32) und in einer 64 Bit Version (IA64) ausgeliefert. Die Unterschiede werden in den folgenden Punkten jeweils gegenübergestellt:

- Speicherverwaltung: Es wird virtuelles Speichermanagement mit Paging verwendet. Jeder Thread erhält den vollen adressierbaren Speicherbereich. Wenn der Speicherplatz nicht mehr ausreicht, werden Seiten anhand einer modifizierten FIFO Strategie ausgelagert. Die

Seitengröße beträgt 4kB bei IA32 und 8kB bei IA64. Es gibt auch die Möglichkeit, größere Speicherseiten zu verwenden. Der Standardwert für große Speicherseiten liegt bei 4MB. Dieser Wert kann in der *Registry* angepasst werden. Die Seiten werden in Auslagerungsdateien ausgelagert, deren maximale Größe 16TB bei IA32 und 512TB bei IA64 beträgt. Es können maximal 16 Auslagerungsdateien erstellt werden, die jeweils auf anderen logischen Laufwerken liegen müssen. Die Standardgröße der Auslagerungsdatei beträgt 150% des installierten Hauptspeichers. Der adressierbare Speicher beträgt bei 32 Bit je nach Version (Standard-, Enterprise- oder Datacenter-Server) 4 – 64GB – ermöglicht durch die Adress Windowing Extension (AWE) –, sowie 16TB bei der 64 Bit Version. Der Speicherbereich wird zweigeteilt. Der obere Bereich steht dem Betriebssystem selbst zur Verfügung und wird als Kernelspeicher bezeichnet. Der untere Bereich steht Anwendungen zur Verfügung und wird Anwendungsspeicher genannt. Der Kernelspeicher kann bei der 32 Bit Version zugunsten des Anwendungsspeichers verschoben werden (Editieren der Datei *boot.ini*). Zum Schutz des Betriebssystems vor Abstürzen gibt es zwei zusätzliche Speicherpools, den Paged Pool-Speicher und den Non-Paged Pool-Speicher. Beide Pools sind für Kernelprozesse reserviert, damit diese immer genügend Speicher zur Verfügung haben. Der Paged Pool Bereich kann ausgelagert werden, der Non-Paged Pool nicht.

- Prozessverwaltung: Unter Windows besteht jeder Prozess zumindest aus einem Thread. Windows 2003 verwendet präemtives Multitasking um CPU-Taktzyklen zwischen laufenden Threads aufzuteilen, es wird auch Multithreading unterstützt. Die Menge an Zeit, die einem Thread zur Verfügung steht, ist bei Workstations kurz und bei Servern lang (je nach dem, ob die Optimierung von Dialog- oder Hintergrundprozessen im Vordergrund steht). Dies kann in der *Systemsteuerung* verändert werden. Jeder Thread hat eine Basispriorität. Diese kann aber nicht direkt beeinflusst werden, sondern über so genannte Prioritätsklassen – Low (Priorität 4), Normal (Priorität 8), High (Priorität 13) und Realtime (Priorität 24) –, und über Prioritätsebenen – Abovenormal (Priorität 9) und Belownormal (Priorität 7). Der Scheduler arbeitet nun die Prioritätsliste von der höchsten Priorität beginnend ab. Warten mehrere Threads auf einer Stufe, werden sie der Reihe nach abgearbeitet. Damit Threads niedrigerer Priorität nicht „verhungern“, gibt es eine dynamische Komponente. Wenn ein Thread zu Gunsten eines Threads höherer Priorität übergangen wird, wird die Priorität des niederwertigeren Threads erhöht, bis er Systemressourcen erhält. Wenn ein Thread mit hoher Priorität die CPU eine Zeit beansprucht hat, wird seine Priorität verringert (Tanenbaum, 2002, S.858f). Die Basispriorität kann beim Start einer Anwendung oder im *Task Manager* beeinflusst werden. Wenn ein Prozess auf Realtime gesetzt wird, hat dieser eine höhere Priorität als der Prozess, der Tastatureingaben überwacht. Falls dieser Prozess mit Priorität Realtime nicht sauber läuft, muss das System neu gestartet werden, damit wieder auf eine Eingabe reagiert wird. Prozesse sollten also nie, oder nur aus sehr gutem Grund, auf Priorität Realtime gesetzt werden. Dies gilt im Übrigen auch für Linux und i5/OS.

- Ein-/Ausgabeverwaltung: Aus Platzgründen sei hier nur der *Logical Disk Manager (LDM)* erwähnt, der es ermöglicht, Festplatten zu konfigurieren. So können z.B. übergreifende Volumes (Verknüpfung von freien Speicherbereichen auf mehreren Festplatten), aber auch RAID-Systeme (mit Ausnahme von RAID0+1 und RAID10) softwaremäßig konfiguriert werden. Die Performance von Hardware RAIDs kann durch die Speichertreiber von Windows gesteigert werden, da sie es ermöglichen, die Stripegröße auf Hardware RAID-Controllern, denen des Betriebssystems anzupassen.

Linux

In diesem Abschnitt erfolgt ein kurzer Einblick in die Besonderheiten von Linux Betriebssystemen. Weiterführende Hinweise finden sich in (Wielsch, 1999; Tanenbaum, 2002). Linux wird in den unterschiedlichsten Distributionen ausgeliefert. Von Semiramis unterstützt werden momentan SUSE Linux Enterprise Server und Red Hat Enterprise Server. Alle diese Distributionen haben ihre spezifischen Unterschiede, deshalb wird hier nur sehr allgemein auf Besonderheiten eingegangen. Da Linux eine Open Source Software ist, werden immer wieder neue Features entwickelt, die leicht in das System eingebunden werden können. Deshalb ist hier auch keine pauschale Aussage zu treffen.

- Speicherverwaltung: Wie auch unter Windows wird virtuelle Speicherverwaltung verwendet. Jeder Prozess auf einer 32 Bit Maschine erhält 3GB Speicher für sich, der Rest ist für Kernelprozesse reserviert. Die Seitengröße ist fix, abhängig vom Prozessor, z.B. 4kB beim Intel. Unter Linux gibt es die Möglichkeit spezielle Auslagerungspartitionen (swap Partitionen) anzulegen, oder aber auch bis zu acht Auslagerungsdateien fester Größe zu verwenden. Das Verwenden von swap Partitionen ist aus Performancegründen zu bevorzugen. Der auszulagernde Teil des Hauptspeichers sollte doppelt so groß sein wie der reale Speicher.

- Prozessverwaltung: Unter Linux stehen Prozesse in einer hierarchischen Ordnung. Jeder Prozess hat eine eindeutige Nummer – die Process Identification (PID) –, und die Nummer seines Elternprozesses. Wie auch bei Windows wird präemtives Multitasking verwendet, das auf Basis von Threads arbeitet. Multithreading wird ebenfalls unterstützt. Für das Scheduling unterscheidet Linux drei Klassen von Prozessen: Realtime FIFO, Realtime Round Robin und Timesharing. Realtime FIFO Prozesse haben höchste Priorität und können nicht unterbrochen werden, außer von einem anderen Realtime FIFO Prozess. Realtime Round Robin Prozesse haben gleiche Priorität wie Realtime FIFO Prozesse, nur dass sie durch eine Uhr unterbrochen werden können. Timesharing Prozesse sind die Standardklasse. Jeder Thread hat eine Scheduling-Priorität, deren Wert zwischen 1 und 40 liegt. Zusätzlich wird jedem Thread ein Quantum zugeordnet, das ist die Anzahl der Uhrticks, die er noch laufen darf. Während der Ausführung wird dieses Quantum pro Uhrtick um eins verringert. Der Scheduler entscheidet, welcher Prozess als nächstes an die Reihe kommt, anhand der Güte. Die Güte errechnet sich aus der Priorität plus dem Quantum. Der Prozess mit der größten Güte wird ausgewählt (Realtime Prozesse bekommen einen Aufschlag von 1000 auf die Güte). Ist das Quantum auf 0, wird dem

Prozess die CPU entzogen. Die Priorität eines Prozesses kann durch den *nice* Systemaufruf verändert werden.

- Ein-/Ausgabeverwaltung: Auch von Linux wird softwaremäßiges RAID unterstützt. Je nach Distribution werden RAID0, RAID1 und RAID5 unterstützt. Auch festplattenübergreifende Volumes können erstellt werden.

i5/OS

IBM iSeries-Systeme (früher AS/400) und das installierte Betriebssystem i5/OS sind als Applikationsserver konzipiert, und unterscheiden sich dadurch von so genannten „PC-Betriebssystemen“ wie Windows oder Linux. i5/OS läuft nur auf IBM Hardware mit einem Power 5 Prozessor. Das System ist objektbasiert, d.h. alles was vom System verwendet wird – auch eine Datei – ist ein Objekt. Die Objektorientierung geht hier aber wesentlich weiter, wie das bei Windows der Fall ist. Ein Objekt kann nur über seine Methoden angesprochen und verändert werden. Dadurch wird erhöhte Sicherheit und Integrität erreicht, da nur definierte Methoden ein Objekt verändern können. Dies gilt auch für Benutzer mit Adminstratorrechten. Eine relationale Datenbank (DB2 UDB for iSeries) ist fest im Betriebssystem integriert. Es gibt eine Vielzahl an Besonderheiten, hier sei auf (Bedernjak, 2005; KTW, 2006a; Wiki, 2006) verwiesen. In dieser Arbeit werden die Begriffe i5 oder iSeries zur Bezeichnung der Plattform und i5/OS zur Bezeichnung des Betriebssystems verwendet.

- Speicherverwaltung: Die iSeries bietet einen durchgängig beschreibbaren Adressraum, Single Level Storage (ein großer virtueller Speicher, zusammengefasst aus Festplatten und Hauptspeicher) genannt, der aufgrund der 64 Bit Adressierung 2^{64} Bytes adressieren kann. Applikationsprogramme arbeiten mit Objekten und sprechen diese direkt mit ihrem Namen an, nicht mit ihrer Adresse. Das Betriebssystem kümmert sich im Hintergrund um das Ein- und Auslagern dieser Objekte in den Hauptspeicher. Objekte werden auch hier seitenweise ein- und ausgelagert.
- Prozessverwaltung: Der Power 5 Prozessor von IBM ist ein 64 Bit RISC Prozessor und unterstützt Multithreading. Eine Besonderheit bei der Prozessverwaltung ist die Verwendung von Subsystemen und Pools. Jobs sind Prozesse auf dem System. Jeder Job ist einem Subsystem zugeordnet. Jeder Job besteht aus einem oder mehreren Teiljobs (Threads). Pools sind Arbeitsspeicherbereiche. Subsysteme sind mindestens einem Pool zugeordnet. Jeder Job in einem Subsystem läuft genau in einem dem Subsystem zugeordneten Pool. Jedes Subsystem bekommt einen gewissen – einstellbaren – Anteil an den CPU’s des Systems. Der Scheduler verwendet so genannte Prioritätsbänder. Es gibt 8 Bänder und 100 Prioritäten, wobei hier eine kleine Zahl eine hohe Priorität bedeutet. Ein Band besteht wiederum aus mehreren Prioritäten, z.B. beinhaltet das Band High Priority die Prioritäten 1-9, das Band 1 die Prioritäten 10-16, oder das Band 7 die Prioritäten 90-99. Der Scheduler geht nun folgendermaßen vor: Zuerst werden alle Jobs im High Priority Band abgearbeitet. Hat jeder Job in diesem Band eine Zeitscheibe lang die CPU beansprucht, so wird wieder von vorne gestartet. Diesmal werden aber die Bänder High Priority und Band 1 der Reihe nach abgearbeitet. Haben alle Jobs in beiden Bändern eine

Zeitscheibe lang die CPU beansprucht, wird wieder von vorne gestartet und die Bänder High Priority, Band 1 und Band 2 abgearbeitet, usw. Die Konsequenz daraus ist, dass Prozesse im High Priority Bereich nur dem System vorbehalten sind, da sie andere Prozesse blockieren können, und Prozesse in höheren Bändern „verhungern“ können. Es empfiehlt sich daher, Band 1 für Applikationen zu verwenden (KTW, 2006a).

- Ein-/Ausgabeverwaltung: Applikationsprogramme auf einer iSeries sind aufgrund des Technology Independent Machine Interfaces (TIMI) unabhängig von der darunter liegenden Hardware. Hardware wird mittels des System Licensed Internal Code (SLIC) angesprochen. Aufgrund des Prinzips des Single Level Storage weiß ein Anwender nie, auf welcher Festplatte seine Daten abgespeichert sind. Das Betriebssystem kümmert sich im Hintergrund darum, wo die Daten abgespeichert werden, und sorgt für eine gleichmäßige Auslastung der Festplatten. i5/OS unterstützt alle Arten von RAID-Systemen. Eine Besonderheit ist auch der so genannte Input Output Prozessor (IOP). Dieser ist für die Verwaltung der Hardware-Steckkarten zuständig und entlastet somit den Hauptprozessor. Der IOP ist wesentlich leistungsfähiger als ein DMA-Controller in einer Intel Architektur. IBM geht aber in Zukunft aufgrund steigender Prozessorleistung und schnellerer Bussysteme wieder dazu über, Steckkarten anzubieten, die keinen IOP benötigen und direkt über den Hauptprozessor verwaltet werden. Eine iSeries unterstützt logische Partitionen. So können verschiedene Betriebssysteme, wie AIX oder Linux, gleichzeitig gebootet werden, und diesen Festplatten- und Prozessorressourcen zugeteilt werden.

Weiterführende Hinweise zu Betriebssystemen finden sich in (Brause, 2001; Silberschatz, 2005).

2.2.2 Datenbanken

In diesem Kapitel erfolgt ein kurzer Einblick in die Architektur von Datenbank-Managementsystemen (DBMS). Abbildung 3 zeigt alle gängigen Komponenten eines DBMS. Die für eine Performanceanalyse wichtigsten Komponenten, welche sich in den drei weiter unten vorgestellten Datenbank-Plattformen wieder finden, werden nachfolgend beschrieben (Vossen, 2000, S. 31ff).

- Optimierer: Der Optimierer verändert Statements, so dass sie mit bestmöglicher Effizienz ausgeführt werden können, ohne jedoch das Ergebnis zu verändern. Die drei weiter unten beschriebenen Datenbank-Plattformen verwenden einen Cost Based Optimizer (CBO). Ein CBO erstellt Statistiken für Tabellen und Indizes und berechnet unter Berücksichtigung der aktuellen Systemauslastung, die Kosten für die einzelnen Aktionen des aktuell erstellten Zugriffsplans (siehe unten). Es werden mehrere Zugriffspläne erstellt, und der mit den niedrigsten Kosten ausgewählt. Daher kommt auch der Begriff „teure“ SQL-Anweisung. Eine Anweisung, für die der CBO hohe Kosten ermittelt, wird als „teuer“ bezeichnet.
- Zugriffsplanerstellung und Code-Erzeugung: Es wird überprüft, ob auf die benötigten Daten Zugriffsstrukturen (Indizes, siehe dazu Abschnitt 2.2.2.1) vorhanden sind, und ein möglichst effizienter Zugriffsplan bzw. ein Ausführungsprogramm erstellt. Jeder Auftrag

lässt sich im Wesentlichen auf Lese- oder Schreiboperationen auf einem Primär- oder Sekundärspeicher zurückführen, und genau diese Folge wird vom DBMS erstellt.

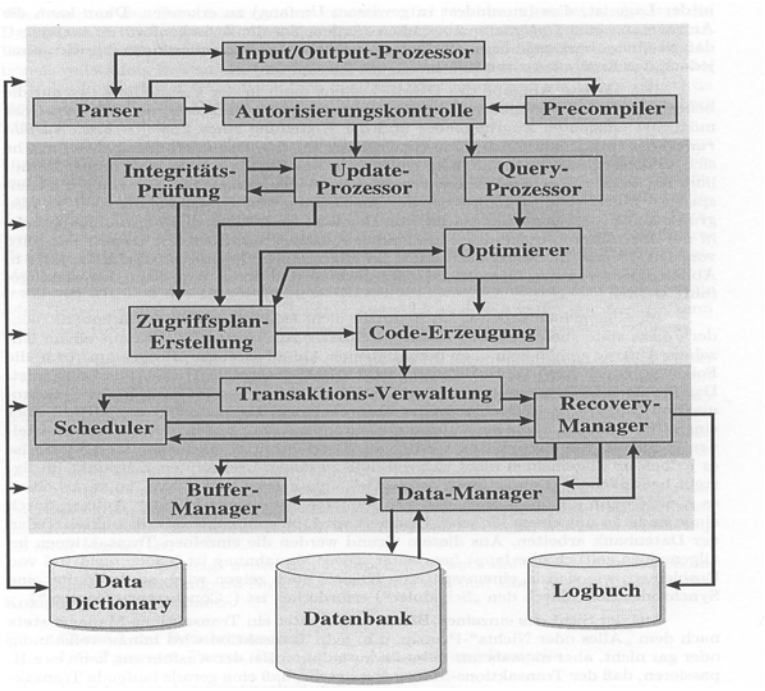


Abbildung 3 – Komponenten eines DBMS (Quelle: Vossen, 2000, S. 33).

- Transaktionsverwaltung: Im Allgemeinen arbeiten mehrere Benutzer mit einer Datenbank. Der durch den oben beschriebenen Ablauf erzeugte Code – beispielsweise ein *UPDATE* Statement – wird als Transaktion bezeichnet. Die Aufgabe der Transaktionsverwaltung besteht darin, diese quasi parallel ablaufenden Transaktionen zu synchronisieren und zu verhindern, dass z.B. zwei parallele Transaktionen die gleichen Daten bearbeiten. Der Transaktionsmanager arbeitet nach dem „Alles oder Nichts“ Prinzip, d.h. eine Transaktion wird immer vollständig oder gar nicht ausgeführt. Dieses „Alles oder Nichts“ Prinzip nennt sich auch atomare Transaktion.

- Recovery-Manager: Wenn eine Transaktion nicht vollständig zu Ende gebracht wird, ist es Aufgabe des Recovery-Managers, die Daten wieder in den ursprünglichen Zustand zu bringen. Auch wenn das System Hard- oder Softwarefehler macht, oder wenn es abstürzt, ist es Aufgabe des Recovery-Managers, das System wieder anlaufen zu lassen, und in einen konsistenten Zustand zurückzuversetzen.

- Buffer-Manager: Dient der Verwaltung des Hauptspeichers. Da Zugriffe auf Sekundärspeicher eine wesentlich längere Zeit beanspruchen, ist das DBMS bestrebt, möglichst viele Daten im Hauptspeicher zu behalten. Befinden sich Log-Einträge im Buffer, werden diese so schnell wie möglich auf den Sekundärspeicher zurück geschrieben. Daten werden aus Performancegründen in bestimmten Intervallen – so genannten Checkpoints – auf die Sekundärspeicher geschrieben. So sind oft so genannte „dirty Pages“ im Buffer vorhanden, das sind Daten, die geändert wurden, aber noch nicht auf die Festplatte geschrieben wurden. Finden Checkpoints oft statt, wird die Performance des

laufenden Systems negativ beeinflusst. Laufen sie selten, wird beim Hinunterfahren des Systems bzw. beim Wiederanlaufen des Systems nach einem Absturz eine wesentlich längere Zeit benötigt. Beim Hinunterfahren des Systems müssen die dirty Pages auf die Festplatte geschrieben werden. Beim Wiederanlaufen des Systems müssen alle Log-Einträge, die noch nicht auf Festplatte geschrieben wurden, wiederhergestellt werden (Log-Rollback).

- Logbuch: Änderungen werden immer zuerst in das Logbuch geschrieben, bevor sie physisch auf die Datenbank geschrieben werden. Dies ist Grundvoraussetzung für das Funktionieren des Recovery-Managers. Aufgrund der Wichtigkeit des Logbuches einerseits, und andererseits aus Performancegründen – Logs werden selten gelesen – ist es empfehlenswert, dieses auf eine andere Festplatte zu speichern als das Datenfile, welches häufiger geschrieben wird.

2.2.2.1 Indizes

Der Zugriff auf einzelne Datensätze, insbesondere das Suchen nach bestimmten Datensätzen, kann durch Indizes wesentlich beschleunigt werden. Ein Index ist vergleichbar mit einem Stichwortverzeichnis für ein Buch. Ein Index ist eine zusätzliche Struktur, welche meist für ein bestimmtes Attribut, die aktuell in der Tabelle vorkommenden Werte dieses Attributs zusammen mit der Adresse des jeweiligen Datensatzes enthält. Das Suchen eines bestimmten Datensatzes läuft in zwei Schritten: zuerst wird der Index gesucht, und aus diesem die Adresse des gesuchten Datensatzes ermittelt (Vossen, 2000, S. 445).

Die Suche nach einem Datensatz ist dadurch viel schneller, da nur der Index durchsucht werden muss, und nicht die ganze Tabelle. Das Schreiben eines Datensatzes ist hingegen langsamer, da zusätzlich zum Datensatz der Index aktualisiert werden muss. Bei der Anlage von Indizes muss also immer ein Kompromiss zwischen gewünschter Schreib- und Lesegeschwindigkeit gefunden werden. Beispielsweise macht es für Stammdaten – diese werden oft gelesen und selten geändert – Sinn, einen Index zu vergeben, und für Bewegungsdaten – diese werden öfter geschrieben als gelesen – kann man durch einen Index möglicherweise einen negativen Effekt erzielen. Einen Kompromiss stellt die Verwendung eines Clustered Index dar. Mit diesem werden die Werte eines Attributs, die nahe beieinander liegen in benachbarten Speicherplätzen abgespeichert. Dadurch ist das Schreiben schneller als bei einem herkömmlichen Index, das Lesen dauert hingegen nur unwesentlich länger (Vossen, 2000, S. 507).

Nachfolgend einige Arten von Zugriffen auf Indizes am Beispiel des Oracle DBMS (Hoermann, 2006). Diese Zugriffsarten finden sich im Ausführungsplan wieder, siehe Abschnitt 3.3.4.2.

- Full Table Scan: Ist kein Index vorhanden, wird ein Full Table Scan durchgeführt. Dies ist der ineffizienteste Zugriff auf Daten.
- Index by RowiId: Der Zugriff auf eine Spalte einer Tabelle nach einem vorgelagerten Indexzugriff.

- Index Unique Scan: Wenn auf eine Spalte in einer *WHERE* Bedingung ein Index liegt, wird ein Index Unique Scan durchgeführt. Dieser ist effizient für den Zugriff auf einen Datensatz.
- Index Range Scan: Wenn in der *WHERE* Bedingung ein Index für eine Treffermenge größer eins gefunden wird. Der Index Range Scan ist effizient für wenig Datensätze.
- Index Full Scan: Dieser ist effizient wenn eine sortierte Ergebnismenge benötigt wird.
- Fast Full Index Scan: Effizient, wenn alle Daten (*WHERE* und *SELECT*) im Index vorhanden sind.

Aus der Auflistung sieht man, dass man je nach Art der SQL-Abfrage die Verwendung einer bestimmten Indexzugriffsart erzwingen kann. Hat man beispielsweise eine SQL-Abfrage mit mehreren *WHERE*-Bedingungen, so kann durch die Vergabe von Indizes auf alle Daten der *SELECT* und *WHERE* Bedingung ein Fast Full Index Scan erzwungen werden. Weiters muss beachtet werden, ob die Abfrage auf Bewegungsdaten zielt. In diesem Fall ist von der Indexerstellung abzusehen, da der erhöhte Aufwand beim Schreiben der Daten den positiven Effekt beim Lesen aufheben kann.

2.2.2.2 Caching

Wie schon erwähnt ist ein DBMS bestrebt, möglichst viele Daten aus Performancegründen im Hauptspeicher zu behalten. Dieser Bereich, in dem die Daten abgelegt werden, wird Buffer genannt. Die Verwaltung des Buffers ist ähnlich einer virtuellen Speicherverwaltung, mit folgenden spezifischen Aspekten (Vossen, 2000, S. 441f):

- Wenn der Buffer voll ist, muss ein Block ausgelagert werden. Ein Betriebssystem verwendet dazu z.B. die LRU Strategie. Bei Datenbanken ist dies oft nicht optimal, weil oft Blöcke zur temporären Berechnung herangezogen werden, und danach nicht mehr benötigt werden. Es kann hier also unter bestimmten Bedingungen zusätzlich eine MRU Strategie sinnvoll sein.
- Ein Betriebssystem kann auszulagernde Blöcke meistens einfach überschreiben. Bei Datenbanken muss immer geprüft werden, ob Daten verändert wurden (dirty Pages). Wenn dies der Fall ist, muss der Block in den Sekundärspeicher zurück geschrieben werden.

Auch ist es sinnvoll, für Stamm-, Bewegungs- und Customizingdaten unterschiedliche Strategien zu verwenden. Bewegungsdaten wachsen in der Regel schnell und werden selten gelesen, deshalb sollte man diese nicht cachen. Stammdaten werden selten geschrieben aber oft gelesen, deshalb macht es Sinn diese zu cachen. Customizing Daten sollten immer gecached werden.

2.2.2.3 Partitionierung

Mit Partitionierung kann man eine Tabelle in mehrere logische Segmente teilen. Durch diese Möglichkeit kann die Lesegeschwindigkeit für Aggregatfunktionen erhöht werden. Wenn zum Beispiel eine Tabelle in Monaten partitioniert ist, muss eine SQL-Abfrage, die eine Aggregatfunktion über ausgewählte Monate berechnet, nur die Partitionen, die die relevanten

Daten enthalten, abfragen. Aber auch das Schreiben der Daten ist schneller, da aufeinander folgende Daten auch in benachbarten Speicherbereichen abgelegt werden und kein Index benötigt wird. Partitionierung ist mit einem Clustered Index vergleichbar. Eine weitere Möglichkeit ist es, Tabellen über mehrere Festplatten zu partitionieren. Dadurch kann durch parallele Lesevorgänge eine Verbesserung der Geschwindigkeit erzielt werden (Whalen, 2005, S. 84f).

2.2.2.4 Sperren

Datensätze, aber auch Indizes, die gerade von einer Transaktion bearbeitet werden, müssen für andere Transaktionen gesperrt werden. Gesperrte Datensätze sind für andere Operationen nicht verfügbar. Dazu gibt es nicht exklusive Sperren (Lesesperren), d.h. andere Operationen können ebenfalls Lesesperren auf dieses Objekt setzen. Eine *SELECT* Abfrage beispielsweise verwendet eine nicht exklusive Sperre. Andere Operationen dürfen die Daten lesen, sie dürfen aber nicht beschrieben werden. Weiters gibt es exklusive Sperren (Schreibsperren). Hier können andere Operationen keine Sperren mehr auf das betreffende Objekt setzen. Exklusive Sperren werden für *INSERT*, *UPDATE* oder *DELETE* verwendet. Sperren können je nach Möglichkeiten und Konfiguration des DMBS eine Tabelle bzw. einen Index, eine Speicherseite oder einen Datensatz betreffen (Vossen, 2000, S. 555ff; Delany, 2001, S. 773ff). Treten Sperren häufig auf, wird die Performance eines Systems negativ beeinflusst.

Zu erwähnen sind auch optimistische und pessimistische Sperrverfahren (Härder, 2001, S. 442). Beim pessimistischen Sperren ist paralleles Arbeiten in Echtzeit auf ein und derselben Datenquelle (Tabelle, Seite oder Datensatz) nicht möglich, da hier nur ein Anwender zu einer gegeben Zeit die Sperre besitzt. Erst nach der Freigabe der Daten kann ein anderer darauf zugreifen und eine Sperre setzen. Beim optimistischen Verfahren ist paralleles Arbeiten zwar möglich, aber nur auf lokalen Kopien der Daten. Bearbeiten zwei Benutzer parallel dieselben Daten, so kann der Benutzer, der seine Daten als Erster speichert, diese Aktion erfolgreich durchführen. Der langsamere Benutzer verliert dadurch seine geänderten Daten.

Der Zustand von Transaktionen, bei dem mindestens zwei Transaktionen auf Betriebsmittel warten, die einer anderen beteiligten Transaktion zugeteilt sind, nennt man Verklemmung oder Deadlock. Beim Auftreten einer Verklemmung werden alle beteiligten Transaktionen blockiert. Um die Blockierung aufzulösen verwendet ein DBMS unterschiedliche Verfahren, wie z.B. das Timeout-Verfahren. Dieses beendet Transaktionen, sobald deren Wartezeit auf eine Sperre eine festgesetzte Schranke überschreitet. Ein DBMS kennt aber auch Strategien, um Deadlocks erst gar nicht entstehen zu lassen. Siehe dazu (Härder, 2001, S. 436). Falls das DBMS keine Möglichkeit findet, Deadlocks zu beseitigen, kann der Eingriff eines Administrators erforderlich sein, der einen der beteiligten Prozesse beendet.

2.2.2.5 Besonderheiten der einzelnen Datenbank-Plattformen

Oracle Database 10g

In diesem Abschnitt erfolgt ein kurzer Einblick in die Besonderheiten der Oracle Datenbank. Weiterführende Hinweise finden sich in (Whalen, 2005). Die physische Datenstruktur eines Oracle Systems besteht unter Anderem aus:

- Einem oder mehreren Datenfiles: Hier werden die Daten und Indizes gespeichert. Im Unterschied zur DB2 UDB for iSeries werden in einem Datenfile mehrere Tabellen gespeichert.
- Einem oder mehreren Controlfiles: Hier wird die Struktur der Datenbank gespeichert, wie z.B. der Name der Datenbank Instanz, oder wo sich die Daten- und Redo-Logfiles befinden.
- Zwei oder mehreren Redo-Logfiles: Im Redo-Log werden alle Änderungen gespeichert, siehe Abschnitt 2.2.2.
- Optional Archive Logfiles: Redo-Logs werden periodisch überschrieben. Um ein System zu jedem Zeitpunkt wiederherstellen zu können, ist es notwendig, die Redo-Logs zu sichern, in so genannte Archive Logfiles.
- Einem Parameterfile (init.ora).

Datenfiles werden in Tablespaces logisch zusammengefasst. Redo-Logfiles werden in Redo-Log Gruppen zusammengefasst.

Ein Oracle Datenbankserver besteht aus der Datenbank und der Oracle Instanz, die in Abbildung 4 schematisch dargestellt ist. Die Oracle Instanz besteht aus den Hintergrundprozessen und der System Global Area (SGA). Die Hintergrundprozesse heißen Database Writer (DBWR), Log Writer (LGWR), Process Monitor (PMON), System Monitor (SMON), Checkpoint (CKPT) sowie weitere in der Grafik nicht angeführte Prozesse, wie etwa der Archiver. Die Bedeutung dieser einzelnen Prozesse wurde teilweise in Abschnitt 2.2.2 beschrieben und kann detailliert in (Whalen, 2005, S.55ff) nachgelesen werden.

Für diese Arbeit verdient die SGA eine detailliertere Betrachtung. Die SGA liegt im Hauptspeicher und ist eine Shared Memory Region, die Oracle verwendet, um Daten und Control Informationen für eine Instanz zu speichern. Seitenersetzung geschieht nach einer modifizierten LRU Strategie. Die SGA besteht unter Anderem aus:

- Database Buffer: Hier werden die am meisten verwendeten Datenblöcke gespeichert.
- Redo Log Buffer: Hier werden die Redo Einträge gespeichert.
- Library Cache: Hier werden Shared SQL-Statements gespeichert. Der Ausführungsplan für jedes SQL-Statement ist hier abgelegt. Wenn mehrere Applikationen dasselbe Statement verwenden wollen, steht es ihnen zur Verfügung. Aus Performancegründen ist es deshalb sinnvoll, *stored procedures* zu verwenden, da diese leichter wieder verwendbar sind. In

einer s*tored procedure* wird eine Kette von SQL-Anweisungen unter einem Namen zusammengefasst. Durch Verwendung von Variablen in *WHERE*-Bedingungen wird bessere Wiederverwendbarkeit erreicht. Die Verwendung von *stored procedures* ist in allen drei beschriebenen Datenbank Plattformen zu empfehlen.

- Shared Pool: Hier befindet sich der Shared Bereich des Library Cache und der Data Dictionary Cache.

Zu erwähnen sind hier auch noch der Oracle Listener, der auf Benutzeranfragen wartet, und der Serverprozess, der die Benutzeranfragen ausführt. Oracle verwendet einen Cost Based Optimizer, ein Rule Based Optimizer wird aus Rückwärts-Kompatibilitätsgründern unterstützt.

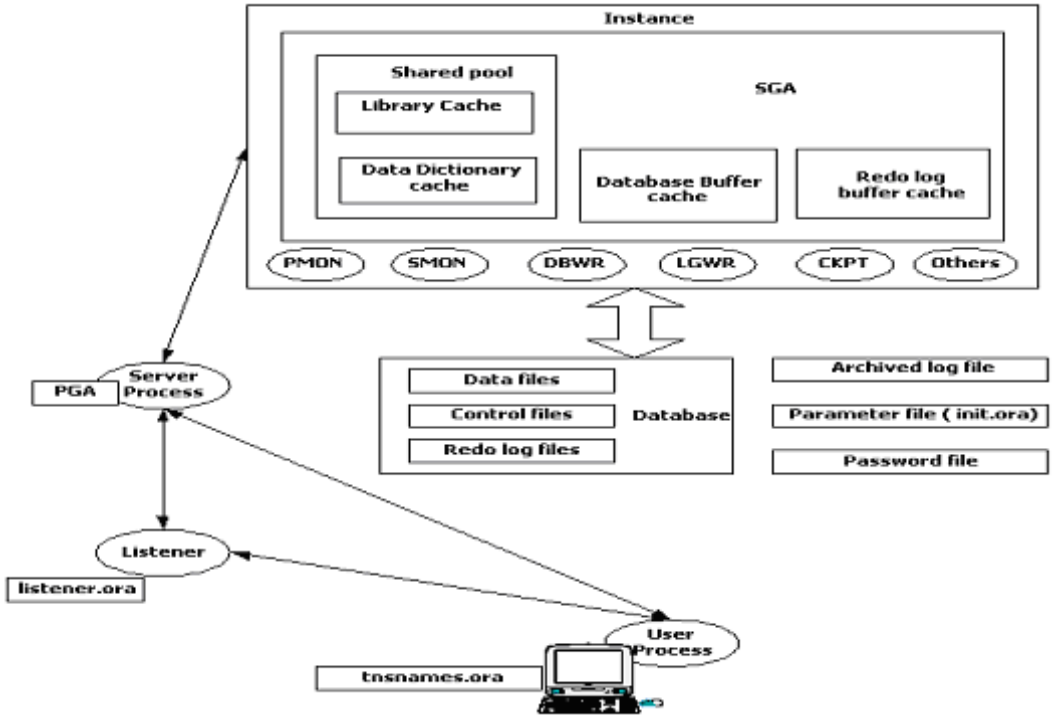


Abbildung 4 – Die Oracle Instanz (modifiziert nach: Oracle, 2006).

DB2 UDB for iSeries

In diesem Abschnitt erfolgt ein kurzer Einblick in die Besonderheiten der DB2 UDB for iSeries. Weiterführende Hinweise finden sich in (Bedoya, 2004; KTW, 2006a). Wie schon oben erwähnt ist die DB2 UDB for iSeries fest in das Betriebssystem i5/OS integriert. Dies ergibt einige Vorteile in der Verwaltung, da Betriebssystem-Werkzeuge auch für die Administration der Datenbank verwendet werden können. Auch bei der Benutzerverwaltung ergeben sich Vorteile, da Benutzer nur einmal erstellt werden müssen. Die Verwaltung der physischen Dateien und der Buffer obliegt dem Betriebssystem, wie in Abschnitt 2.2.1 dargestellt. Im Unterschied zu Oracle und SQL-Server existiert für jede Tabelle, Index oder View eine eigene Datei.

SQL Begriff	i5/OS Begriff
Table	Physical file
View	Non-keyed logical file
Index	Keyed logical file
Column	Field
Row	Record
Schema	Library, Collection
Log	Journal
Isolation Level	Commitment control level

Abbildung 5 – SQL und iSeries Begriffe (modifiziert nach: Bedoya, 2004, S. 6).

Da die iSeries vor dem SQL Standard entwickelt wurde, gibt es einige Besonderheiten bei den Bezeichnungen, siehe Abbildung 5.

Anhand Abbildung 6 ergibt sich eine wesentliche Besonderheit: die Verwendung von zwei verschiedenen Optimierern, der Classic Query Engine (CQE) und der SQL Query Engine (SQE). Die SQE – weil moderner und performanter – soll die CQE in Zukunft ablösen. Momentan werden aber je nach Releasestand mehr oder weniger SQL-Statements vom Query Dispatcher zur SQE geleitet. Statements, die noch nicht vom SQE bearbeitet werden können sind z.B. *INSERT WITH VALUES* Statements oder länderspezifische Sortierungen, die von Semiramis oft verwendet werden. Der CQE bietet die Möglichkeit, Encoded Vector Indizes (EVI) zu verwenden, welche auf *WHERE* Klauseln mit wenigen Ergebnissen optimiert sind.

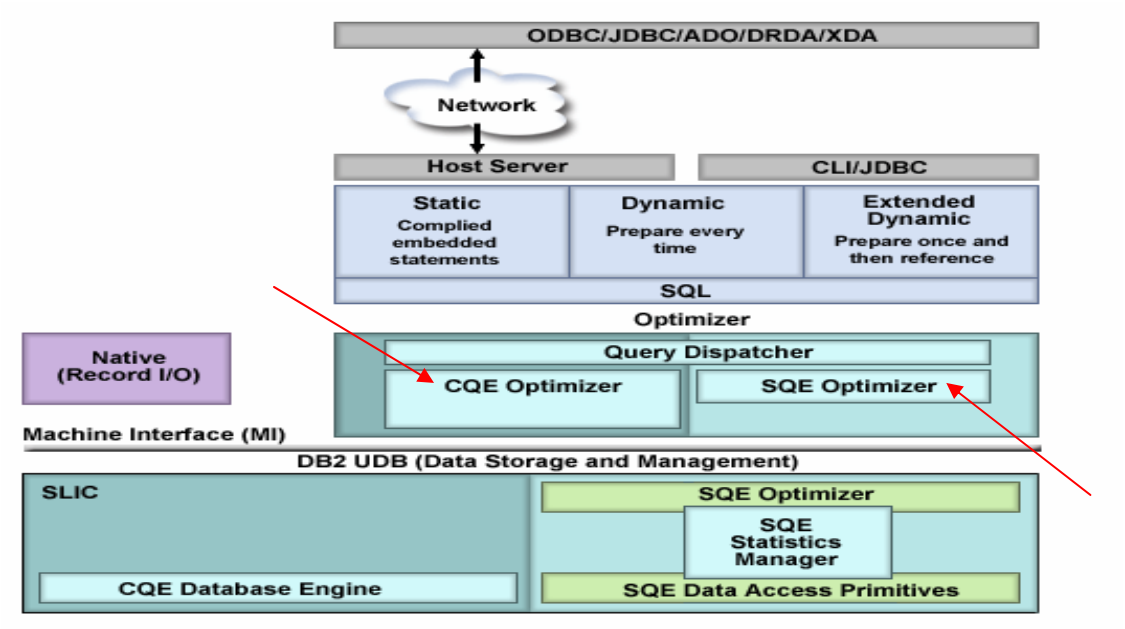


Abbildung 6 – DB2 UDB for iSeries spezifische Implementierung (Quelle: KTW, 2006a).

MS SQL-Server 2000 bzw. MS SQL-Server 2005

In diesem Abschnitt erfolgt ein kurzer Einblick in die Besonderheiten des MS SQL-Server. Weiterführende Hinweise finden sich in (Delany, 2001; Bauder, 2006). Die Architektur des MS

SQL-Server ist in Abbildung 7 schematisch dargestellt. Die SQL-Server Engine kann grob in zwei Bereiche unterteilt werden, der Relational Engine und der Storage Engine. Die Relational Engine ist dafür zuständig, Statements zu parsen, zu optimieren und diese auszuführen. Die Storage Engine enthält Komponenten für den Zugriff und das Ändern von Daten auf Festplatten. Wie Oracle verwendet auch der SQL-Server einen Speicherpool. Dieser ist unterteilt in den Buffer Pool, der Daten und Indizes enthält, und den Procedure Cache, wo *stored procedures* gespeichert sind. Diese Speicherbereiche sind aber nicht statisch, sondern werden dynamisch in ihrer Größe angepasst. SQL-Server kann aber auch direkt auf den Cache des Betriebssystems zugreifen, wenn Seiten mit einer Größe von mehr als 8MB benötigt werden. Die Verwaltung des SQL-Server Buffers kann statisch oder dynamisch geschehen. Bei der dynamischen Variante wird immer ein Bereich von >4MB für das Betriebssystem belassen. Die Seitenersetzung geschieht nach einer modifizierten LRU Strategie, mittels des so genannten Lazywriters, der periodisch den Buffer scannt, und die am längsten nicht verwendeten Seiten auslagert.

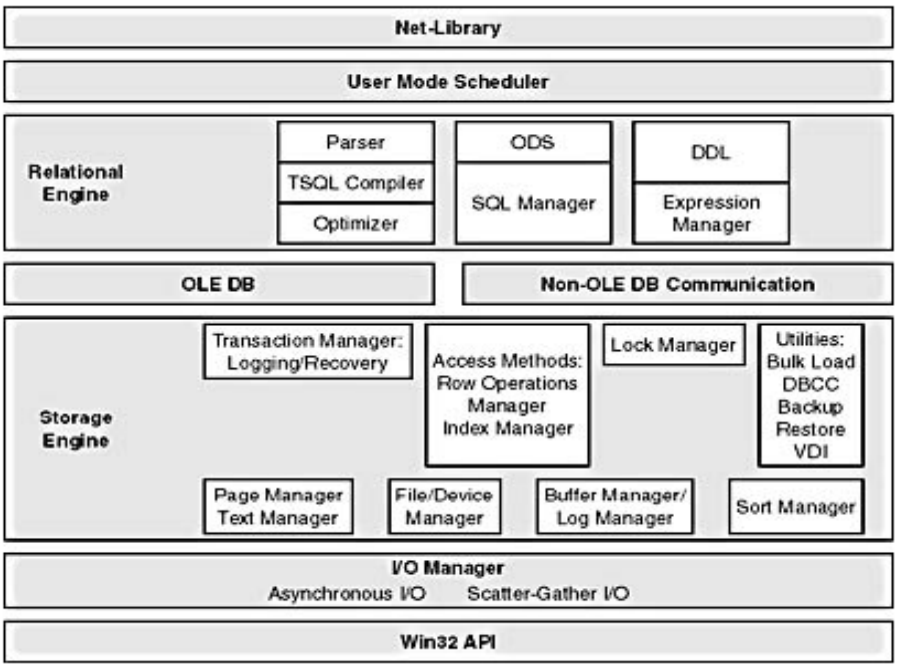


Abbildung 7 – MS SQL-Server 2000 Architektur (Quelle: Delany, 2001, S.71).

Wie Oracle verwendet der SQL-Server Daten- und Logfiles, die mehrere Tabellen enthalten. Im Unterschied zu Oracle werden keine Tablespaces verwendet. Eine Datenbank benötigt ein Primary-Datafile. Weitere Datenfiles werden als Secondary-Datafiles erstellt.

Eine besondere Eigenschaft des Optimierers des SQL-Servers ist Autoparametrisierung. Eine Abfrage wie „*SELECT firstname, lastname, titel FROM employees WHERE employeeID = 6*“ wird vom Optimierer in eine parametrisierte Form umgewandelt, also in „*SELECT firstname, lastname, titel FROM employees WHERE employeeID = @p1*“. So kann dieses Statement für andere Abfragen wieder verwendet werden. Siehe dazu Abschnitt 3.3.4.2.

Weiterführende Hinweise zu Datenbanken finden sich in (Date, 2004; Türker, 2005).

2.2.3 Netzwerke

ERP-Systeme sind in der Regel verteilte Systeme, deshalb kommt deren Vernetzung eine große Bedeutung zu. Eine ausführliche Behandlung dieses Themas würde für diese Arbeit zu umfangreich sein, deshalb sei hier auf (Tanenbaum, 2003; Kurose, 2002) verwiesen. Nachfolgend eine kurze Definition der für den weiteren Verlauf der Arbeit wichtigsten Begriffe.

2.2.3.1 Multi-Tier Architekturen

Die gängigste Architektur einer Web Applikation – oder auch eines webbasierten ERP-Systems – kann anhand Abbildung 8 dargestellt werden.

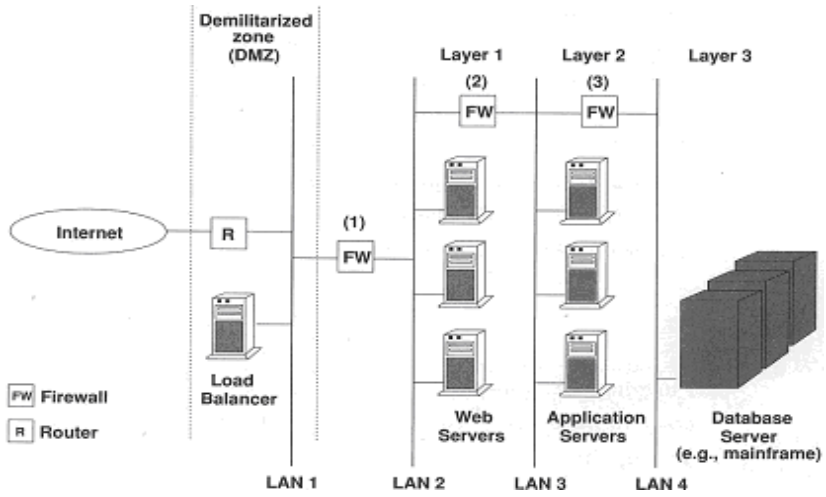


Abbildung 8 – Typische Multi-Tier Website Architektur (Quelle: Menascé, 2002, S.159).

Die erste Schicht – auch Präsentationsschicht genannt – stellt die Schnittstelle zum Benutzer dar. In der zweiten Schicht, der Applikationsschicht, wird die Business-Logik implementiert. In der dritten Schicht, der Datenbankschicht, liegt die Logik für den Datenbank-Zugriff. Durch die Trennung der einzelnen Schichten erlangt man mehr Robustheit, und Stabilität. Auch ist Sourcecode einfacher wartbar, da z.B. bei Änderung des Datenbanksystems nur die Komponenten der dritten Schicht verändert werden müssen.

2.2.3.2 Ethernet

Ethernet ist das gängigste Protokoll für Local Area Network (LAN) Netzwerke. Die am Netz angeschlossenen Geräte teilen sich eine Leitung. Da nur jeweils ein Gerät die Leitung verwenden kann, wird vor der Datenübertragung überprüft, ob bereits eine andere Übertragung stattfindet (Trägerprüfung bzw. carrier sense). Findet eine Übertragung statt, blockiert der Sender die Leitung, um alle anderen Sender zu alarmieren. Diese Blockierung wird Kollision genannt. Dann zieht er sich zurück und wartet eine zufällige Zeitspanne bis zum nächsten Versuch. Tritt eine zweite Kollision ein, dann wird die Zufallswartezeit verdoppelt, usw. Damit werden die konkurrierenden Übertragungen auseinander gezogen, und der erste Sender der die Leitung in Anspruch nehmen wollte, hat die Gelegenheit, als Erster zu senden (Tanenbaum, 2003, S. 84ff). Ethernet wird mit

Übertragungsraten von 10 Mbit/s bis 10 Gbit/s realisiert (IEEE, 2006). Die Anzahl der Kollisionen beeinflusst natürlich die Übertragungsrate der Leitung negativ.

2.2.3.3 VPN

Ein Virtuelles Privates Netzwerk (VPN) ist ein privates Netzwerk, das die öffentliche Kommunikationsverbindung (Internetanbindung) eines Unternehmens nutzt, und Privatsphäre und Sicherheit durch Tunnelprotokolle und Sicherheitsprozeduren herstellt. Ein VPN bietet ähnliche Möglichkeiten wie eine Standleitung, nur zu wesentlich niedrigeren Kosten. VPN kann dazu verwendet werden, ein ERP-System von einer Außenstelle zu erreichen. Es gibt drei verschieden Arten von VPN:

- Secured VPN: In einem solchen VPN wird der Netzwerkverkehr verschlüsselt.
- Trusted VPN: Hier versichert der Provider dem Kunden, dass kein Anderer die ihm zugewiesene Verbindung nutzt.
- Hybrid VPN: Die Kombination aus den beiden oben genannten VPN.

Zum Thema VPN siehe auch (Böhmer, 2005).

2.2.3.4 MTU

Jedes Netzwerk besitzt eine Maximum Transfer Unit (MTU). Die MTU gibt das größte Paket an, das vom Netzwerk übertragen werden kann ohne fragmentiert zu werden. Ist ein empfangenes Paket größer als die MTU eines anderen Netzwerkes, muss es für die Übertragung in kleinere Fragmente zerlegt werden (Hunt, 2003, S.18). Die MTU wird durch die verwendete Technik bzw. das verwendete Protokoll bestimmt. So hat z.B. Ethernet eine MTU von 1500 (IEEE, 2006). Die Path-MTU beschreibt die Paketgröße, die entlang der gesamten Wegstrecke übertragen werden kann, ohne fragmentiert werden zu müssen. Jedes Teilnetzwerk bzw. jeder Router erhöht den Overhead eines Paketes, und die MTU wird immer kleiner. So kommt z.B. bei einem Paket, welches über einen VPN-Tunnel übertragen wird, ein zusätzlicher Overhead hinzu, und die MTU reduziert sich. Um die Performance zu steigern, empfiehlt es sich, die Path-MTU für einen Pfad zu ermitteln und genau diesen Wert für die maximale Paketgröße zu verwenden (Conrads, 2004, S. 234). Siehe dazu Abschnitt 3.3.6.1.

2.2.3.5 HTTPS

Das Hyper Text Transfer Protocol Secure (HTTPS) Protokoll wurde von der Firma Netscape entwickelt und ermöglicht eine gesicherte Verbindung zwischen Rechnern (Neumann, 2002). HTTPS verschlüsselt die Daten, die über das HTTP Protokoll übertragen werden, und wird zum gesicherten Zugriff auf einen Webserver verwendet. Im Browser wird „*https://...*“ angegeben. Die Verbindung wird üblicherweise über Port 443 hergestellt. Um einen Client zu authentifizieren, werden X.509 Zertifikate verwendet. Zertifikate sind Dateien, die Informationen über die Servermaschine und den öffentlichen Schlüssel des Servers enthalten (Böhmer, 2005). Während des Authentifizierungsprozesses, der Secure Socket Layer (SSL)-Handshake genannt wird, wird das Server Zertifikat zum Browser des Clients übertragen. Wenn eine zentrale

Authentifizierungsstelle dieses Zertifikat mit ihrem öffentlichen Schlüssel unterzeichnet hat, vertraut der Client diesem Zertifikat, und die Verbindung wird hergestellt (Bragg, 2004, S. 614). Zum Thema Netzwerksicherheit siehe auch (Panko, 2004).

2.2.3.6 Hubs, Switches, Router

Hubs, Switches und Router sind essentielle Bestandteile von Netzwerken, die Einflüsse auf die Performance haben können. Hier gilt aber auch wie bei Hardware-Controllern und Bussystemen, dass eine detaillierte Betrachtung für diese Arbeit zu ausführlich wäre. Es sei wiederum auf (Tanenbaum, 2003; Kurose, 2002) verwiesen.

2.2.4 Java Virtual Machine

Die Java Virtual Machine (JVM) stellt die Schnittstelle zwischen ausführbaren Java Klassen und dem jeweiligen Betriebssystem dar. Sie ist somit die Grundlage für die Plattformunabhängigkeit der Programmiersprache Java. Die JVM wurde von der Firma SUN Microsystems spezifiziert. Es gibt verschiedenste Implementierungen unter anderem von SUN selbst – für die Betriebssysteme Windows, Solaris, Linux und MacOS – oder die eigene Implementierung von IBM für i5/OS und AIX. Um die Performance der JVM analysieren zu können, müssen zunächst der Java Heap, die Garbage Collection (GC) und der Just in Time Compiler (JIT) näher betrachtet werden.

Abbildung 9 zeigt eine schematische Darstellung des Java Heaps. Wenn ein neues Objekt erzeugt wird, kommt es in den Young Generation Bereich. Wenn der Platz nicht mehr zur Erzeugung neuer Objekte ausreicht, wird eine Garbage Collection durchgeführt. Der Speicherplatz für nicht mehr benötigte Objekte wird freigegeben, noch referenzierte Objekte werden in den Tenured (oder Old) Generation Bereich verschoben. Wenn der Tenured Generation Bereich zu voll ist, wird eine Full Garbage Collection durchgeführt. Die Full GC defragmentiert den Speicher, sie verursacht ein so genanntes „stop the world“, d.h. es werden alle anderen Threads blockiert, und das System steht für diese Zeit still. Die Full GC sollte daher nicht zu lange dauern, damit sie vom Benutzer nicht wahrgenommen wird. Zusätzlich zu diesen Speicherbereichen gibt es noch den Permanent Space, wo die Klassen in den Hauptspeicher geladen werden. Der Heap einer 32 Bit JVM ist theoretisch bis 4GB adressierbar. Aufgrund der Zweiteilung des Speichers für System- und Benutzerprozesse, den viele Betriebssysteme verwenden, halbiert sich dieser Bereich. Um Platz für andere Prozesse auf dem System zu schaffen, ergibt sich ein realistischer Wert von <1,5GB für den Java Heap. Um möglichst viele Objekte im Speicher zu behalten, und damit die Performance einer Anwendung zu verbessern, sollte der Heap so groß wie möglich sein. Zu bedenken ist jedoch die größere CPU Auslastung aufgrund erhöhter Garbage Collection bei einem großen Heap (Johnson, 2005; CIS, 2006d).

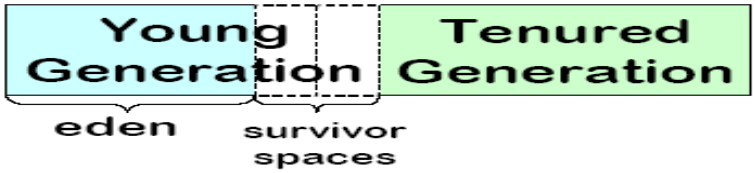


Abbildung 9 – Java Heap (Quelle: Johnson, 2005).

Die so genannte Hot Spot Implementierung der JVM bietet für die Garbage Collection die Optionen „Parallel Collection“ für die Young Generation und „Concurrent Collection“ für die Old Generation an. Beide Varianten sind nur für Mehrprozessorsysteme sinnvoll und werden in Semiramis verwendet. Die Standard Collection verwendet so viele Threads für die GC, wie Prozessoren auf dem System vorhanden sind. Mit der Option Parallel Collection kann die Anzahl der Threads für die GC eingeschränkt, und so der Overhead für die Synchronisation vieler GC Threads reduziert werden. Der Concurrent Collector läuft dauernd im Hintergrund und versucht so, Systemstillstände zu vermeiden. Die gesamte Zeit für die GC und die CPU Auslastung wird aber durch die Verwendung der Concurrent Collection erhöht (Johnson, 2005).

Standardmäßig werden Java-Programme jedes Mal interpretiert, wenn sie aufgerufen werden. Der Just in Time (JIT) Compiler kompiliert Java Programme, so müssen sie beim nächsten Aufruf nicht mehr interpretiert werden. Dadurch wird das System beim Starten der JVM langsam, je länger es aber läuft, desto performanter wird es. Der Speicherverbrauch wird durch die Verwendung von JIT erhöht (Shirazi, 2003, S. 76). Um das System beim Start zu entlasten, werden Programme standardmäßig erst beim zweitausendsten Aufruf kompiliert.

2.2.5 JDBC-Treiber

Java Database Connectivity (JDBC) besteht aus einem Satz Application Programming Interfaces (API’s) und Spezifikationen, die es einer Java Anwendung ermöglicht, SQL-Abfragen auf einer relationalen Datenbank auszuführen (Shirazi, 2003, S. 463). Semiramis verwendet eine eigene Implementierung des JDBC-Treibers, den Semiramis Open Database Connectivity (ODBC)-Treiber. Die Semiramis ODBC-Treiber sind für jede Datenbank- bzw. Betriebssystem Plattform gesondert implementiert, und können die Ursache eines Performanceproblems sein. Es wird hier aber nicht detailliert darauf eingegangen.

2.2.6 Semiramis

Die Beschreibung der Semiramis-Architektur wurde aus (CIS, 2004a) übernommen und wird in Abbildung 10 dargestellt. Semiramis ist vollständig in Java implementiert. Die Semiramis-Architektur ist webbasiert und besteht aus drei Schichten (siehe Abschnitt 2.2.3.1): der Bedienungsoberfläche, der Anwendungslogik und der Datenhaltung. Als Bedienungsoberfläche wird der Microsoft Internet Explorer verwendet. Für die Anwendungslogik wurde ein eigener Web- und Applikation-Server, der Semiramis Application-Server (SAS) entwickelt. Ein SAS kann auf Microsoft Windows 2000 und 2003 Server Systemen, auf SUSE Linux Enterprise Server und Red Hat Enterprise Server, sowie auf IBM iSeries Systemen installiert werden. Um Lastverteilung und Ausfallssicherheit zu erreichen, können mehrere SAS auf unterschiedlichen Rechnern im Netzwerk installiert werden. Der Message Server, der häufig auf dem Rechner läuft, auf dem das DBMS installiert ist, synchronisiert die einzelnen SAS untereinander. Die dritte Schicht umfasst die Datenhaltung. Sie besteht aus herkömmlichen DBMS. So werden Oracle, Microsoft SQL-Server und DB2 UDB for iSeries unterstützt.

Die Anwendungslogik übermittelt Ausgabeaufträge an den Semiramis Output Manager (SOM). Der SOM übernimmt die Ausgabe von Berichten, die entweder gedruckt, gefaxt, oder als E-Mail

versendet werden können. Um Berichte als Email versenden zu können, werden Dateien erzeugt (pdf, html, doc, xls, usw.), die als Anhang dem Email hinzugefügt werden. Der SOM kann nur auf Windows 2000 oder 2003 Server Systemen installiert werden. Zur Erstellung und Ausgabe von Berichten wird die Software Crystal Reports der Firma Business Objects verwendet (Crystal, 2006).

Die Kommunikation zwischen Clients und Server bzw. zwischen den einzelnen SAS und dem SOM erfolgt über das HTTPS Protokoll. Die Anmeldung eines Benutzers am System erfolgt über Zertifikate.

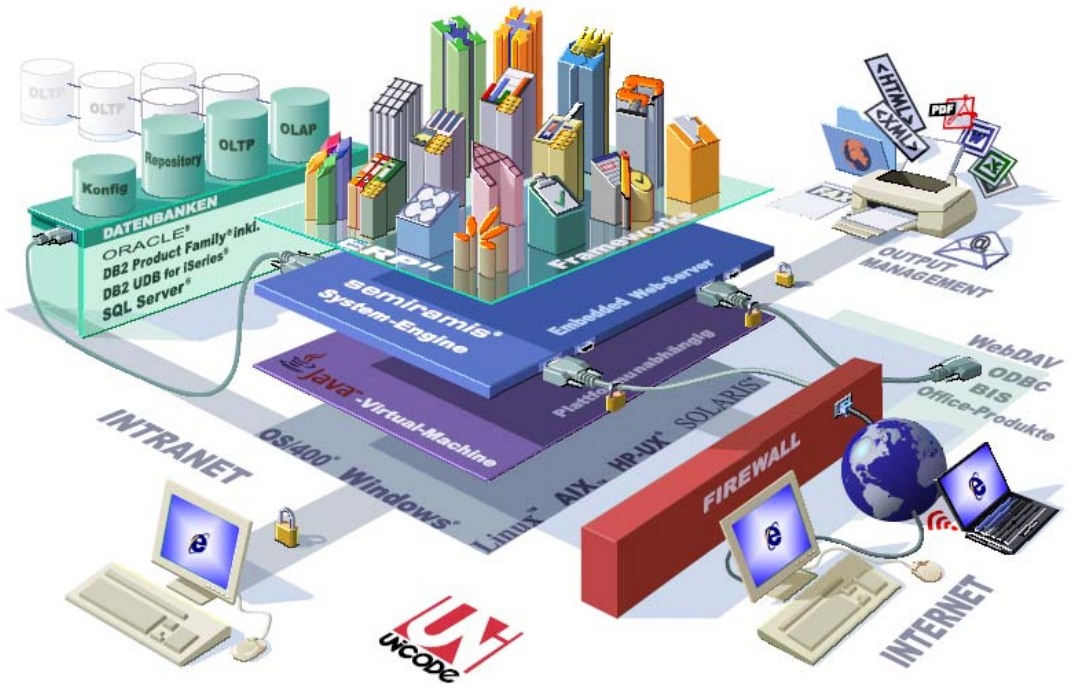


Abbildung 10 – Semiramis Architektur (Quelle: CIS, 2004b).

Die betriebswirtschaftlichen Funktionen werden über so genannte Frameworks abgebildet. Die einzelnen Frameworks heißen Basis, Vertrieb, Beziehungs-Management, Workflow-Management, Dokumenten-Management, Beschaffung, Lagerlogistik, Produktion, Disposition, Software-Entwicklung und System-Management. Innerhalb der Frameworks befinden sich die Anwendungen. Eine Anwendung ist ein ausführbares Programm und unterstützt mehrere Aktionen, wie z.B. das Laden oder Neuanlegen einer Instanz.

Zu erwähnen sind hier auch die Begriffe Business Object und Business Entity. Diese werden in Semiramis etwas anders gehandhabt, als man es von der Objektorientierung bzw. der Datenmodellierung her kennt. Business Objects enthalten ausschließlich Daten. Unter weitestgehender Berücksichtigung der dritten Normalform werden Ausschnitte der Realität modelliert. Business Objects sind im Wesentlichen die Abbildung der Datenbanktabellen durch den Semiramis ODBC-Treiber, wie z.B. Tabellen für Länder, Anreden oder Titel. Da dadurch eine

Vielzahl an Business Objects entsteht, werden sie zu Business Entities zusammengefasst, um eine anschauliche betriebswirtschaftliche Größe herzustellen. Business Entities sind z.B. Artikel oder Kunden. Die Identifizierung eines Business Objects geschieht nicht durch einen Primärschlüssel, sondern durch einen Global Unique Identifier (GUID).

Zur Datenhaltung verwendet Semiramis die folgende logische Partitionierung der Datenbanken:

- Die Systemkonfigurations-Datenbank enthält Informationen über die Systemtopologie und Benutzer. Ein Semiramis-System ist meist über mehrere Systeme verteilt, deshalb wird diese Datenbank einmal an zentraler Stelle benötigt.
- Die Repository-Datenbank enthält Entwicklungsobjekte, Javaklassen, Metadaten und Berichte. Auch wichtige Systemprotokolle finden sich hier.
- Die Online Transaction Processing (OLTP)-Datenbank enthält Stamm-, Bewegungs- und Customizingdaten. Innerhalb eines Semiramis-Systems können mehrere OLTP-Datenbanken vorhanden sein, um Daten getrennt halten zu können.
- Die Online Analytical Processing (OLAP)-Datenbank enthält Statistikdaten in Form eines Data Warehouses.

Die unterschiedlichen Datenbankhersteller verwenden verschiedene Dialekte des SQL Standards. Um die Kommunikation zu vereinheitlichen, wurde eine Object Query Language (OQL) implementiert, die auf objektorientierten und relationalen Konzepten basiert. Die Semiramis OQL ist sehr stark an den SQL Standard angelehnt. Sie hat den Vorteil, dass sie für alle von Semiramis unterstützten DBMS verwendet werden kann. Die Semiramis OQL hat aber den Nachteil, dass nicht alle Funktionen des DBMS genutzt werden können.

Der „Motor“ eines Semiramis-Systems ist die so genannte System Engine. Eine in einer JVM-Instanz gestartete System Engine stellt einen SAS dar. Die System Engine stellt unter Anderem einen embedded Webserver, Dienste für das Cache Management, das Thread- und Session Management, sowie den Sperr-, Persistenz- und den Transaktionsdienst zur Verfügung. Semiramis übernimmt also – vor allem aus Performancegründen – Funktionen, die von einem DBMS durchgeführt werden könnten.

Um einen schnellen Zugriff auf Daten sicherzustellen, ist in Semiramis ein Cache Management implementiert. Das Caching-Verhalten von Business Objects ist, wie unter Abschnitt 2.2.2.2 für das Datenbank Caching beschrieben, unterschiedlich für Stamm-, Bewegungs-, oder Customizingdaten. Folgende Strategien werden von Semiramis angeboten:

- Alle Instanzen: Lädt bei der ersten Verwendung des Business Objects alle Instanzen dieses Objekts. Diese Strategie ist für Customizing Daten sinnvoll.
- Die Standardstrategie LRU: Hier werden nur die zuletzt benutzten Objekte gecached. Diese Strategie sollte bei Stammdaten verwendet werden.
- Nicht cachen: diese Strategie empfiehlt sich für Bewegungsdaten.
- Permanent: Diese Business Objects befinden sich immer im Cache.

Es ist auch möglich, den Cache in mehrere Partitionen einzuteilen. So können z.B. Stamm- und Customizing Daten in eine Partition abgelegt werden, und Bewegungsdaten in eine andere. Durch diese Strategie werden häufig benutzte Stammdaten auch dann nicht von Bewegungsdaten verdrängt, wenn durch eine einzelne Anfrage innerhalb von kurzer Zeit viele Bewegungsdaten erzeugt werden. Durch eine Kombination der erwähnten Strategien kann die Trefferrate des Caches wesentlich erhöht werden. Auch Datenbankanfragen werden vom Cache beeinflusst. Semiramis erstellt Datenbankabfragen unterschiedlich, bzw. ist keine Anfrage notwendig, je nachdem ob ein Objekt im Cache liegt oder nicht. Semiramis verwendet *Prepared Statements*, diese müssen nur einmal vom Optimierer der Datenbank übersetzt werden, und stehen dann anderen Anwendungen zur Verfügung.

Semiramis stellt einen Business Integration Service (BIS) zur Verfügung, und kann über eine VPN-Verbindung von einer Außenstelle erreicht werden.

2.3 Performancekennzahlen und Begriffe

Es folgt nun eine Definition von Begriffen zur Performanceanalyse. Laut (Jain, 1991) gibt es drei verschiedene Resultate, die ein Service Request erzielen kann. Entweder wird er verarbeitet – korrekt oder inkorrekt –, oder er wird aufgrund eines Systemausfalls nicht verarbeitet. Diese Arbeit konzentriert sich auf den ersten Fall, d.h. korrekt verarbeitete Requests.

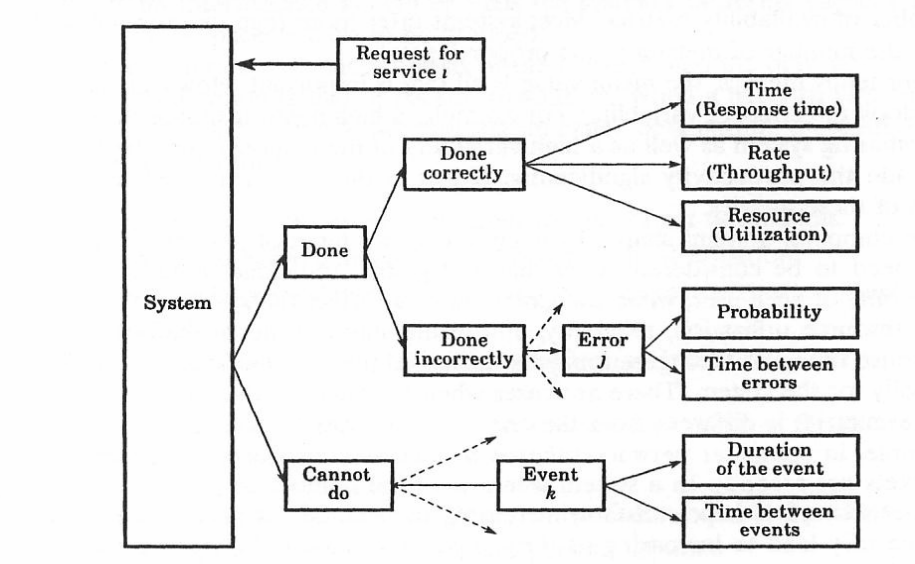


Abbildung 11 – Drei verschiedene Resultate eines Service Requests (Quelle: Jain, 1991, S. 33).

2.3.1 Antwortzeit

Die Antwortzeit wird als die Zeit definiert, die zwischen einem Request und der Antwort des Systems vergeht (Jain, 1991, S. 37). Es können mehrere Zeiten definiert werden, siehe dazu Abbildung 12. Die Reaktionszeit ist die Zeit, die vergeht, bis das System die Verarbeitung startet, also z.B. die Netzwerk-Übertragungszeit. Die Definitionen 1 und 2 der Antwortzeit unterscheiden sich dahingehend, dass bei Definition 2 die Zeit bis zur Übermittlung des letzten Zeichens der

Antwort gemessen wird, während bei Definition 1 die Zeit bis zum Start der Übertragung gemessen wird.

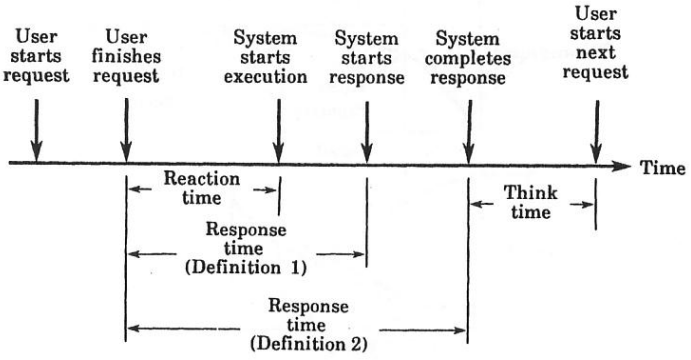


Abbildung 12 – Definition der Antwortzeit (Quelle: Jain, 1991, S. 37).

In (Menascé, 2004) findet sich eine Definition der Antwortzeit bezogen auf Web Applikationen, siehe dazu Abbildung 13.

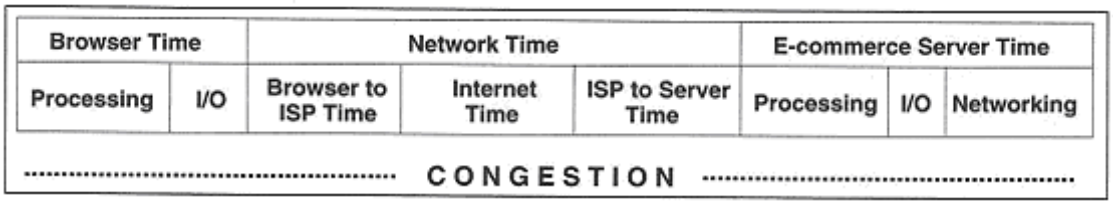


Abbildung 13 – Aufgegliederte Antwortzeit (Quelle: Menascé, 2004, S.14).

Die Browserzeit beinhaltet die Zeit, die Anfrage zu senden und das Ergebnis am Browser darzustellen. Die Netzwerkzeit ist die Zeit, die der Request vom Browser bis zum Server benötigt (Im Falle eines ERP-Systems fällt hier keine Internet Service Provider (ISP)-Zeit an, außer es wird eine Verbindung über VPN hergestellt). Die E-Commerce Serverzeit beinhaltet die Zeit, die der E-Commerce Server, also beispielsweise das ERP-System, benötigt, den Request zu verarbeiten, inklusive allfälliger Verarbeitungs-, I/O-, Netzwerk- und Datenbankzeiten.

Die Roundtrip Time, die im Semiramis-Leistungsmonitoring gemessen wird, entspricht der E-Commerce Serverzeit. Es muss also bei der Analyse eines Semiramis-Systems beachtet werden, dass zu einer gemessenen Antwortzeit noch zusätzliche Zeiten hinzukommen, wie z.B. die Browserzeit.

Antwortzeiten von Systemen sollten so kurz wie möglich sein, um vom Benutzer nicht als störend empfunden zu werden. Es ist natürlich stark von der Art der Applikation und der Arbeitsweise des Benutzers abhängig, welche maximalen Antwortzeiten akzeptiert werden. Eine Richtlinie der akzeptablen Antwortzeiten unterschiedlicher Aktionen in Semiramis findet sich in (CIS, 2006c). Das Hauptaugenmerk dieser Arbeit liegt auf der Analyse bzw. der Verbesserung von Antwortzeiten.

2.3.2 Durchsatz

Der Durchsatz wird definiert als die Rate, mit der Requests von einem System verarbeitet werden können. Für Beispiele von Durchsatz-Kennzahlen siehe Abbildung 14.

System	Durchsatz Kennzahl
OLTP-Datenbank	Transaktionen pro Sekunde (tps)
Netzwerk	Bits pro Sekunde (Mbit/s)
Webserver	HTTP(s) Requests/s
Router	Pakete pro Sekunde (PPS)
CPU	Millionen Instruktionen pro Sekunde (MIPS) Floating Point Operationen pro Sekunde (MFLOPS)
Festplatte	I/O's pro Sekunde Datentransferrate in KB/s

Abbildung 14 – Beispiele für Durchsatz-Kennzahlen (modifiziert nach: Menascé, 2004, S.13).

Der Zusammenhang zwischen Antwortzeit und Durchsatz und die folgenden Begriffe können durch Abbildung 15 veranschaulicht werden (Jain, 1991, S. 38f).

- Nominal capacity oder Bandbreite: Der erreichbare Durchsatz unter optimalen Bedingungen, z.B. die angegebene Bandbreite einer Netzwerkverbindung, 100Mbit/s.
- Wie man aus Abbildung 15 sieht, kommt es bei Erreichen der Nominalkapazität oft zu einer inakzeptablen Antwortzeit. Deswegen ist hier die Usable capacity interessant, bei der eine noch vertretbare Antwortzeit vorliegt.
- Die Knee capacity ist der Punkt, wo die Antwortzeit markant zu steigen beginnt, aber mit größerer Last kein wesentlich größerer Durchsatz mehr erreicht wird. Die Knee capacity ist der erstrebenswerte Zustand, da hier die Ressource am effizientesten genutzt wird.
- Das Verhältnis von Usable capacity zu Nominal capacity wird als Efficiency oder Effizienz bezeichnet. Es gilt somit: *Efficiency = Usable capacity / Nominal capacity.*

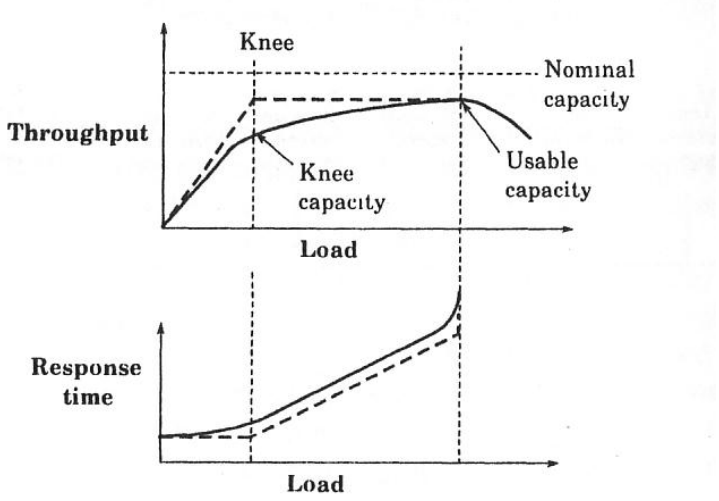


Abbildung 15 – Kapazität eines Systems (Quelle: Jain, 1991, S. 38).

2.3.3 Auslastung

Die Auslastung einer Ressource (busy time) gibt an, zu wie vielen Zeitanteilen diese aktiv ist. Sie wird berechnet aus dem Verhältnis von aktiver Zeit zu Gesamtzeit einer bestimmten Periode. Dagegen gibt die untätige Zeit (idle time) an, zu welchen Zeitanteilen eine Ressource nicht aktiv ist Beide Zeiten werden üblicherweise in % angegeben (Jain, 1991, S. 39).

Es gilt somit: *Auslastung = busy time / (busy time + idle time)* (Fortier, 2003, S. 340).

2.3.4 Klassifikation von Performancekennzahlen

Performancekennzahlen können nach folgendem Schema klassifiziert werden (Jain, 1991, S. 40):

- Higher is Better (HB): Solche Kennzahlen sollen so groß als möglich sein, z.B. der Durchsatz eines Systems.
- Lower is Better (LB): Es wird erstrebt, diese Kennzahlen so niedrig als möglich zu halten. Ein Beispiel ist die Antwortzeit.
- Nominal is Best (NB): Diese Kennzahl soll weder hoch noch niedrig sein. Ein Wert in der Mitte ist erstrebenswert. Die Auslastung einer CPU ist ein solcher Wert. Wenn die Auslastung zu hoch ist, ist das System überlastet. Wenn sie zu niedrig ist, verschwendet man Ressourcen.

2.4 Monitore

Monitoring-Tools können prinzipiell in zwei verschiedene Arten eingeteilt werden: Hardware und Software Monitore.

- Hardware Monitore sind meist externe Geräte, die elektrische Signale messen. Sie haben den Vorteil, dass sie das System, welches überwacht wird, nicht beeinflussen und im Allgemeinen portabel sind. Der Nachteil besteht darin, dass sie – weil sie nur elektrische Signale messen können – bestimmte softwarespezifische Werte wie z.B. eine CPU Warteschlangenlänge nicht erfassen können. Ein Beispiel eines Hardware Monitors ist ein Netzwerk-Tester (Menascé, 2002, S. 494).
- Software Monitore bestehen aus Routinen, die in der Software eines Systems eingebettet sind, mit dem Zweck Status und Events des Systems aufzuzeichnen. Es werden Performancedaten gesammelt, entweder Daten des Systems (CPU, Speicher, I/O), oder Daten einer Software wie z.B. einer Datenbank. Der Nachteil von Software Monitoren ist, dass sie das System beeinflussen, welches überwacht werden soll. Deshalb muss genau abgewogen werden, wie viele Daten des Systems überwacht werden, oder zu welchen Zeitintervallen die Daten aufgezeichnet werden. Aufgrund ihrer Vielzahl an Möglichkeiten sind Software Monitore aber unverzichtbar für eine Performanceanalyse. Eine weitere Möglichkeit, die von vielen Software Monitoren verwendet wird ist die Auswertung von Logfiles, z.B. das Log eines Web Servers. Die schon vorhanden Daten müssen vom

Monitor nur mehr aufbereitet werden, um sie lesbar zu machen (Menascé, 2002, S. 494ff). Software Monitore bilden den Schwerpunkt dieser Arbeit.

2.5 Systematischer Ansatz einer Performanceanalyse

Der folgende systematische Ansatz einer Performanceanalyse wird in Anlehnung an (Jain, 1991; Whalen, 2005) erstellt, und soll im weiteren Verlauf zur Erstellung des Lösungsansatzes beitragen. Die meisten Performanceprobleme sind individuell, jedoch gibt es Schritte, die immer durchgeführt werden sollen. Diese sind:

- **Definieren der Ziele:** Die Definition eines Zieles ist unerlässlich, da damit der Verlauf der Analyse wesentlich beeinflusst wird. So können Ziele beispielsweise die Ermittlung der durchschnittlichen Antwortzeiten eines Systems, oder die Behebung eines konkreten Performanceproblems sein. Falls es sich um die Behebung eines Performanceproblems handelt, ist es wichtig, eine exakte Beschreibung des Verhaltens zu erhalten. Die Aussage „Alles ist langsam“ ist zuwenig, es müssen hier konkrete Aussagen festgehalten werden, z.B. das Erfassen einer Vertriebsauftragsposition dauert fünf Sekunden, soll aber weniger als eine Sekunde dauern.
- **Beschreibung des Systems:** Der erste Schritt der Untersuchung ist, das Verständnis für das System zu erlangen und die Architektur zu dokumentieren, falls das nicht schon geschehen ist. Danach erfolgt die Dokumentation und Validierung der Systemparameter. Es kann vorkommen, dass durch diesen Schritt schon die Ursache des Problems gefunden wird, da z.B. ein Datenfile der Datenbank zu klein ist, oder ein Buffer zu klein konfiguriert wurde.
- **Monitoren des Systems:** Mittels Monitoring-Tools werden Performancedaten gesammelt. Auch hier ist wichtig, die Ergebnisse zu dokumentieren. Aufgrund der Vielzahl an zu untersuchenden Parametern (Betriebssystem, Datenbanken, Netzwerke, usw.) und der Vielzahl an erhältlichen Monitoring-Tools ist dieser Schritt am wenigsten zu systematisieren, wird aber in Abschnitt 3.3 detailliert dargestellt.
- **Analysieren der Resultate:** Das Ziel dieses Schrittes ist es, in Zusammenhang mit der Zieldefinition die gesammelten Performancekennzahlen zu analysieren, um daraus im nächsten Schritt eine Hypothese abzuleiten. Auch die Analyse von Fehler-Logdateien kann hier Aufschluss auf die Art des Problems geben. Auch dieser Schritt ist schwer zu systematisieren, und wird in Abschnitt 3.3 detailliert dargestellt.
- **Erstellen einer Hypothese:** Das Ziel dieses Schrittes ist es, eine Hypothese über die Ursache des Problems aufzustellen. Die Hypothese kann eine einzelne oder auch mehrere Ursachen beinhalten. Wenn es mehrere Ursachen gibt, dann ist es sinnvoll, diese Schritt für Schritt zu behandeln. So kann z.B. die Ursache des Problems ein fehlender Index sein. Dieser Schritt wird in Abschnitt 3.3 detailliert dargestellt.

- **Implementieren der Lösung:** Eine Lösung, für das in der Hypothese beschriebene Problem, wird nun implementiert, d.h. im Fall des fehlenden Indexes wird dieser erstellt. Liegt das Problem an zu knappen Hardwareressourcen wird beispielsweise zusätzlicher Speicher installiert. Zwei Punkte sind hier sehr wichtig: erstens die Änderung zu dokumentieren, und zweitens immer nur eine Einstellung auf einmal zu ändern.
- **Test und Überwachen der Lösung:** In diesem Schritt wird überprüft, ob die Änderung eine wirkliche Verbesserung gebracht hat. Das System muss also wieder mit Monitoring-Tools überwacht werden. Aber auch Tests mit Benutzern am Produktivsystem können angewendet werden, um die Lösung zu validieren. Es können grundsätzlich drei Ergebnisse entstehen: die Performance hat sich verbessert, verschlechtert, oder es ist keine Änderung erfolgt. Im ersten Fall ist die Analyse erledigt, im zweiten und dritten Fall muss wieder bei Schritt „Analysieren der Daten“ begonnen werden.
- **Präsentieren der Resultate:** Dieser Schritt wird oft vernachlässigt. Das Ergebnis der Analyse muss Entscheidungsträgern, Kunden oder Entwicklern präsentiert werden. Ergebnisse können z.B. sein, dass weitere Hardware angeschafft werden muss, ein Programm umgeschrieben werden muss, aber auch dass das System in Ordnung ist. In diesem Fall benötigt der Kunde ebenso ein Ergebnis, da er üblicherweise Geld für die Analyse bezahlt hat. Es ist wichtig, die Ergebnisse in einer verständlichen, kurzen Form, wenn möglich grafisch zu präsentieren.

3 VERBINDUNG THEORIE – PRAXIS: PERFORMANCEANALYSE

Dieses Kapitel befasst sich mit der Umsetzung der theoretischen Erkenntnisse in die Performanceanalyse eines Semiramis-Systems. Das Kapitel liefert eine Vorgehensweise zur Performanceanalyse, sowie eine Aufstellung relevanter Performancekennzahlen, deren Grenzwerte und Handlungsempfehlungen für die unterschiedlichen Plattformen. Des Weiteren werden Flussdiagramme für Vorgehensweisen zur Identifizierung von Anwendungen und Berichten mit hoher Ausführungszeit mit dem Semiramis-Leistungsmonitoring sowie zur Datenbank-Analyse erstellt.

Der unter Abschnitt 2.5. vorgestellte Ansatz zur Performanceanalyse dient als Basis für die Performanceanalyse eines Semiramis-Systems. Zu Beginn müssen die Ziele der Analyse definiert werden. Anschließend folgt die Dokumentation des Systems und die Validierung der Systemparameter. Die Punkte Monitoren des Systems, Analysieren der Resultate, Erstellen einer Hypothese, Implementieren der Lösung, Test und Überwachung der Lösung werden jeweils anhand konkreter Problemstellungen unter Abschnitt 3.3 behandelt. So wird beispielsweise bei der Identifizierung teuerer SQL-Anweisungen aufgezeigt, mit welchen Monitoring-Werkzeugen diese gefunden und analysiert werden können. Es werden mögliche Ursachen, die zur Hypothesenbildung beitragen können erläutert. Nach Möglichkeit wird gezeigt, wie die Ergebnisse

implementiert und überwacht werden können. Der Präsentation der Ergebnisse wird ein eigener Abschnitt gewidmet.

3.1 Zieldefinition

Wie schon erwähnt, müssen vor dem Beginn der Analyse die Ziele festgehalten werden. Es ist wichtig, vom Kunden (unter Kunde wird hier der Auftraggeber einer Analyse verstanden) eine präzise Beschreibung des momentanen und gewünschten Systemverhaltens zu bekommen. Beispielsweise sollte angegeben werden, zu welchen Zeiträumen inakzeptable Antwortzeiten vorkommen, oder welche Anwendungen bzw. Berichte beteiligt sind. Ein Kunde sollte auch seine IT-Infrastruktur dahingehend prüfen, ob andere Anwendungen (z.B. Backups) das Semiramis-System beeinflussen. Befasst sich ein Kunde mit seinem System kann es sein, dass die Ursache selber gefunden wird. Beispielsweise wird eine große Anzahl von Belegen – wie etwa Auftragsbestätigungen – zu einem bestimmten Zeitpunkt gedruckt und das System wird dadurch langsam. Eine Verbesserung der Performance kann hier leicht durch Verlagern der Ausdrucke auf den Abend oder die Mittagspause erreicht werden. Aber es kann auch der umgekehrte Fall eintreten. Beispielsweise verlangt ein Kunde Antwortzeiten, die nicht realistisch sind. Hier muss dem Kunden klar gemacht werden, dass solche Antwortzeiten nicht möglich sind. Beispielsweise wurde ein altes textbasiertes ERP-System eines Kunden durch Semiramis ersetzt. Der Kunde beklagt sich, dass im neuen System alles viel länger dauert. Die Erklärung ist hier, dass in einem graphischen System Aktionen länger dauern können als in textbasierten Systemen. Das Beschreiben des Systemverhaltens ist aber nicht immer einfach. Ist eine Transaktion sehr langsam, kann das weit reichende Auswirkungen auf das gesamte System haben. So kann es vorkommen, dass die Ursache eines Performanceproblems woanders liegt, als dies vom Benutzer wahrgenommen wird.

3.2 Dokumentation und Validierung der Systemparameter

Bevor mit dem Monitoring begonnen wird, muss die Architektur der Semiramis-Systemlandschaft beschrieben werden. Dazu zählen die jeweiligen Betriebssystem- und Datenbank-Plattformen, sowie die Anzahl der SAS und SOM, und deren Verteilung auf unterschiedliche Server. Hier bietet es sich an, eine Skizze der Systemlandschaft zu erstellen.

Anschließend werden die Systemeinstellungen geprüft. Anleitungen dazu finden sich in (CIS, 2004b; CIS, 2005b; CIS, 2006d). Die Systemeinstellungen eines Semiramis-Systems können im *Systemcockpit*, aber auch anhand von Konfigurationsdateien eingestellt werden. Es sollten aber auch die Konfigurationen der Betriebssysteme, Datenbanken und JVM's mit den jeweiligen Verwaltungswerkzeugen geprüft werden. Auch die Netzwerk-Infrastruktur ist zu prüfen. Möglicherweise wird bei der Kontrolle der Systemeinstellungen schon eine offensichtliche Fehleinstellung gefunden, die das Performanceproblem behebt. Ein Beispiel dazu wäre eine zu hoch eingestellte Anzahl von möglichen Threads in der Job-Queue. Dadurch können zu viele

Hintergrundjobs parallel laufen, und das Gesamtsystem wird so stark belastet, dass für andere Anwendungen keine Ressourcen mehr zur Verfügung stehen.

3.3 Analyse der einzelnen Semiramis Komponenten

Die hier vorgestellte Vorgehensweise wird „top down“ durchgeführt. Zuerst werden die Möglichkeiten ausgenutzt, die Semiramis bietet. In den meisten Fällen wird hier das Problem schon gefunden, deshalb empfiehlt es sich, den top down Ansatz zu wählen. Reichen Informationen nicht aus, oder müssen Datenbanken optimiert werden, so kommen Datenbank-Werkzeuge in Betracht. Für detaillierte Informationen zur JVM gibt es ebenso Tools, die in der Version 1.5 mitgeliefert werden. Hat das Datenbank-Tuning keine Verbesserungen gebracht, so müssen Hardware- und Netzwerkkomponenten analysiert werden. Man kann hier natürlich keine strikte „top down“-Strategie anwenden. So bringt beispielsweise Datenbank-Tuning bei Vorliegen eines gravierenden Hardwareengpasses keine sichtlichen Verbesserungen, und es muss zuerst die Hardware erweitert werden, bevor an ein Tuning gedacht werden kann.

Um alle von Semiramis unterstützten Betriebssystem- und Datenbankplattformen analysieren zu können, wurden drei verschiedene Semiramis-Systeme installiert. Es wurde dabei darauf geachtet, dass die aktuellsten Versionen von Betriebssystem- und Datenbank-Plattformen installiert wurden. Auch Semiramis wurde in der aktuellsten Version – 4.2 – installiert. Die installierten Systeme sind:

- Windows 2003 Server mit einer MS SQL-Server 2005 Datenbank.
- Windows 2003 Server mit einer Oracle 10g Datenbank.
- i5/OS mit einer DB2 UDB for iSeries Datenbank Version V5R4M0.

Um aussagekräftige Resultate zu erhalten, wurden jeweils Lasten auf den Systemen erzeugt. Mit Hilfe des Semiramis Request Recorders wurde das Anlegen eines Vertriebsauftrages mit zugehöriger Ausgabe einer Auftragsbestätigung in einem Skript aufgezeichnet. Der Bericht *Auftragsbestätigung* wurde durch einen Unterbericht erweitert, der ein Kreuzprodukt zweier Business Entities (Partner und Artikel) beinhaltet, und dadurch eine künstlich verschlechterte Ausführungszeit aufweist. Dieses aufgezeichnete Skript wurde dann mit dem Open Source Tool *OpenSTA* auf eine Anzahl von 50 virtuellen Usern vervielfältigt. Für Details zu Semiramis Stabilitätstests und Sizing mit *OpenSTA* siehe (CIS, 2004). Die allgemeine Benutzerdokumentation zu *OpenSTA* findet sich in (OpenSTA, 2006).

Die Erzeugung eines Vertriebsauftrages samt Ausgabe der Auftragsbestätigung wurde deshalb gewählt, weil der Vertriebsprozess einer der Kernprozesse in einem Unternehmen ist. Ein weiterer Grund ist, dass die Anwendung Vertriebsaufträge sehr komplex ist, und somit für das Monitoring ausreichend Daten zur Verfügung gestellt wurden. Außerdem wurde so ein durchgängiges Beispiel für alle Plattformen konstruiert, und Ergebnisse konnten besser untereinander verglichen werden.

Es stand kein System mit Linux als Semiramis Applikation- oder Datenbankserver zur Verfügung. Um trotzdem eine Last auf einem Linux-System zu erzeugen, wurde ein Skript erstellt, das eine

große Anzahl an Dateien mit den Befehlen *gzip* und *tar* packt und wieder entpackt. Das Packen und Entpacken von Dateien benötigt sehr viele Ressourcen von CPU, Speicher und Festplatten. Somit erhielt man auch hier aussagekräftige Daten.

Die aufgezeigten Monitoring-Möglichkeiten begrenzen sich auf native Tools der einzelnen Plattformen.

3.3.1 Zentrales Semiramis-Leistungsmonitoring

Das Ziel einer Performancenalyse mit dem zentralen Semiramis-Leistungsmonitoring sollte sein, Aktionen, die eine lange Ausführungszeit haben, zu finden und zu optimieren. Wie schon in Abschnitt 2.3.1 beschrieben, kann hier keine pauschale Aussage über akzeptable Antwortzeiten getroffen werden. Für Semiramis gelten z.B. für die Ausführungszeit von Berichten oder Anwendungen Antwortzeiten von bis zu fünf Sekunden als akzeptabel (CIS, 2006c). Dies ist jedoch immer sehr kundenspezifisch zu sehen. So kann ein Kunde mit Antwortzeiten von fünf Sekunden zufrieden sein, ein Anderer findet dies als störend. Für Aktionen, die oft am Tag ausgeführt werden, werden lange Antwortzeiten eher als störend empfunden als für Anwendungen, die nur einmal pro Woche ausgeführt werden.

Semiramis bietet drei Möglichkeiten zur Performanceanalyse: Datenbank-Leistungsmonitore und deren Auswertungsmöglichkeiten im *Systemcockpit*, Performance-Berichte (beides neu in Version 4.2) und das Datei-Leistungsmonitoring. Diese drei Möglichkeiten werden in den folgenden Abschnitten näher betrachtet.

3.3.1.1 Datenbank-Leistungsmonitore

Seit der Version 4.2 ist in Semiramis ein Standard-Leistungsmonitor integriert. Der Standard-Leistungsmonitor hat seinen Fokus auf der Datenbank, da Datenbankressourcen teuer und schlechter skalierbar sind als beispielsweise SAS, und daher diese bevorzugt optimiert werden sollten. Der Standard-Leistungsmonitor ist immer aktiv und zeichnet Leistungsinformationen für die letzten drei Tage, drei Wochen und drei Jahre in unterschiedlichen Aggregationsstufen auf. Er protokolliert die Leistungsinformationen standardmäßig pro Tag. Um die aufgezeichneten Tage zu aggregieren, muss die Anwendung *Reorganisationsaufträge* ausgeführt werden. Dabei werden die Leistungsinformationen für die Tage zu den Leistungsinformationen der zugehörigen Kalenderwoche aggregiert. Die Leistungsinformationen der Kalenderwochen werden zu den Leistungsinformationen des zugehörigen Jahres aggregiert. Leistungsinformationen zu Tagen, die älter sind als drei Tage, werden von der Reorganisation nach Übertragung in die Kalenderwochen gelöscht. Analog geschieht das mit den Wochen und Jahren. Die Reorganisation komprimiert somit die gesammelten Leistungsdaten und sollte daher täglich in einem Hintergrundauftrag ausgeführt werden (CIS, 2006e).

Der Standard-Leistungsmonitor erfasst Antwortzeiten des Systems bei Dialogzugriffen (Aktionen, die vom Benutzer ausgeführt werden), Ausgabeaufträgen (Druck-, Email- oder Faxausgaben), ODBC-Zugriffen (Ausgabe von Berichten) und Verarbeitungsaufträgen (Hintergrundanwendungen). Diese Informationen bilden die Grundlage für die erste Analyse der Antwortzeiten. Zusätzlich zu diesem Standard-Leistungsmonitor können in der Anwendung

Leistungsmonitore zwei weitere Datenbank-Leistungsmonitore aktiviert werden, der DatabaseMonitor-DatabaseAnlaysis und der DatabaseMonitor-FullAnalysis. Beide Monitore zeichnen zusätzlich die Datenbankstatements der jeweiligen Aktionen auf und belasten dabei das System stärker als der Standard-Leistungsmonitor. Der DatabaseMonitor-DatabaseAnlaysis zeichnet nur die Operationen *Datenbankanweisung ausführen, Datenbank-Transaktion bestätigen* und *Datenbank-Transaktion abbrechen* auf. Der DatabaseMonitor-FullAnalysis zeichnet zusätzlich zu den Operationen der beiden anderen Datenbank-Leistungsmonitore noch die Operationen *Anmeldung für Dialogzugriff, CisOqlSearchStatement.execute, Datenbankverbindung anfordern, Datenbankverbindung öffnen, Datenbankverbindung schliessen* und *Roundtrip GUI* auf. Die zwei zuletzt genannten Monitore zeichnen Leistungsinformationen für die letzten drei Tage, zwei Wochen und das letzte Jahr in denselben Aggregationsstufen wie der Standard-Leistungsmonitor auf (CIS, 2006).

Hat man mit dem Standard-Leistungsmonitor eine oder mehrere Anwendungen bzw. Berichte identifiziert, die Probleme verursachen, kann ein detaillierterer Datenbank-Leistungsmonitor erstellt werden, der sich nur auf die vorher gefundenen problematischen Aktionen beschränkt, um das System nicht zu sehr zu belasten (CIS, 2006).

Die aufgezeichneten Daten aller drei Datenbank-Leistungsmonitore können im *Systemcockpit* eingesehen werden. Es gibt hier zwei Möglichkeiten, den Typ Applikation-Server und den Typ System:

Typ Application-Server, Karteireiter Leistungsinformationen:

Hier sind transiente Leistungsinformationen pro SAS im Arbeitspeicher in Echtzeit verfügbar. Da das zu häufige Speichern der Leistungsinformationen eine zu hohe Last erzeugen würde, werden neue Leistungsinformationen nur in gewissen Zeitabständen in der Datenbank gespeichert und damit persistent. Bei Hinunterfahren eines SAS werden die Daten immer in der Datenbank gespeichert und gehen somit niemals verloren, außer ein SAS stürzt ab, oder wird unsachgemäß beendet. Die noch nicht gesicherten Leistungsinformationen der letzten maximal 24 Stunden können hier eingesehen werden. Es sind nur die Daten des Standard-Leistungsmonitors verfügbar. Einige Abfragefelder sowie Spalten in der Tabelle sind hier nicht vorhanden, da sie vom Standard-Leistungsmonitor nicht unterstützt werden. Dies sind Application-Server, Datenbankabfragen und Suchen. Auch der Filter *Zeitaufwändige Datenbankanweisungen sortiert nach Summe* ist nicht vorhanden**.** Da diese Daten alle im Hauptspeicher des SAS verfügbar sind, erzeugt eine Abfrage auf diese Daten keine Last an der Datenbank (CIS, 2006b).

Typ System, Karteireiter Leistungsinformationen:

Hier werden persistente Leistungsinformationen angezeigt, die vom Leistungsmonitor in der Repository-Datenbank gespeichert wurden. Sie stehen also erst zur Verfügung, nachdem sie vom System in der Datenbank gespeichert wurden (dies ist aber nicht beeinflussbar), oder eine Reorganisation durchgeführt wurde. Deshalb sollte eine Reorganisation vor dem Start der Analyse ausgeführt werden. Es sind alle aktiven Datenbank-Leistungsmonitore, sowie auch Daten für bis zu

drei Jahre in der Vergangenheit verfügbar. Somit sind auch Langzeitanalysen möglich. Die Anwendung bietet einige vordefinierte Filter, die nachfolgend dargestellt werden. Eine detaillierte Anleitung und eine Beschreibung der Spalten findet sich in (CIS, 2006a).

- Filter *Zeitaufwändige Aktionen sortiert nach Summe*: Hier werden Aktionen vom Typ *Roundtrip performAction* (Antwortzeiten von Anwendungen) dargestellt und nach Summe ihrer Ausführungszeit sortiert. Das Feld Abweichung – gilt auch für alle weiteren beschriebenen Filter – ist die Standardabweichung. Dieser Filter kann für alle drei Datenbank-Leistungsmonitore verwendet werden. Zu Beginn der Analyse sollte aber nur der Standard-Leistungsmonitor verwendet werden, da dieser das System nicht so stark belastet. Im Beispiel der Abbildung 16 wurden keine auffälligen Aktionen gefunden, außer der Ausgabe von Performance-Berichten, die aber natürlich für ein Produktivsystem nicht relevant sind. Die relativ langen Antwortzeiten der Vertriebsaufträge liegen daran, dass es sich um ein Testsystem mit wenigen Ressourcen handelt. Auffällig lang dauernde Aktionen können nun über einen neuen Datenbank-Leistungsmonitor vom Typ DatabaseMonitor-DatabaseAnlaysis oder DatabaseMonitor-FullAnalysis analysiert werden um die zugehörigen Datenbankstatements zu finden. Der neue Datenbank-Leistungsmonitor sollte auf die auffälligen Aktionen eingeschränkt werden.

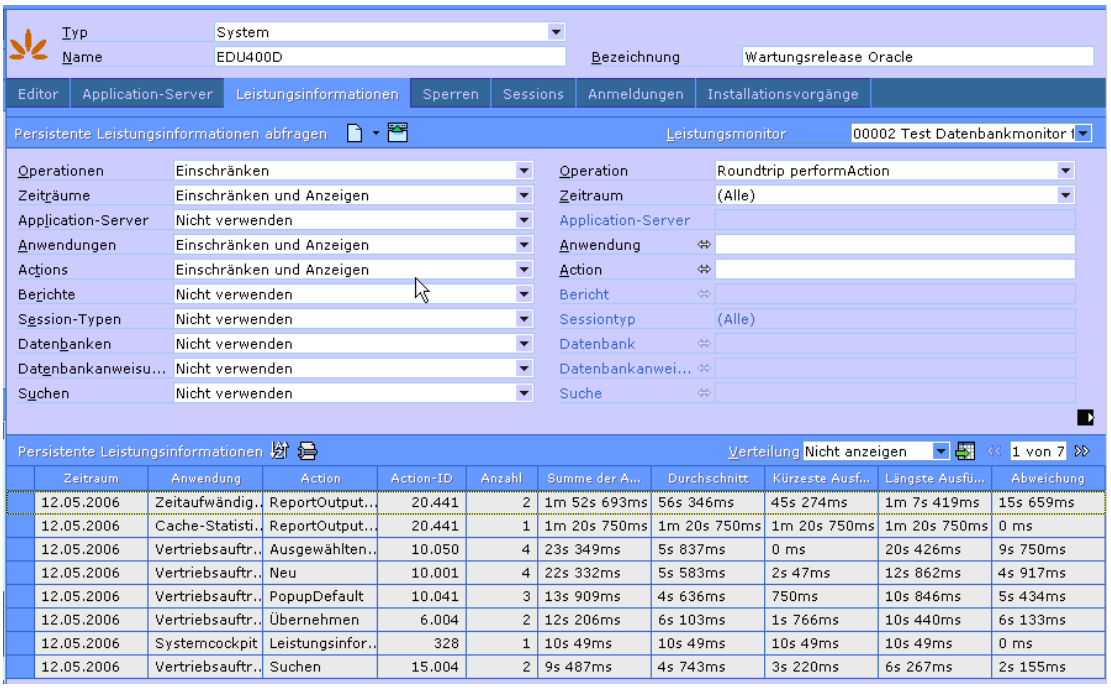


Abbildung 16 – *Zeitaufwändige Aktionen sortiert nach Summe* mit dem DatabaseMonitor-DatabaseAnlaysis.

- Filter *Zeitaufwändige Berichte sortiert nach Summe*: Hier werden Aktionen vom Typ *Roundtrip ODBC-Zugriff* (Aufruf einer ODBC-Funktion während der Berichtsausführung. Ein Bericht kann aus unzähligen ODBC-Aufrufen bestehen) dargestellt und nach Summe ihrer Ausführungszeit sortiert. Dieser Filter kann für alle drei Datenbank-Leistungsmonitore verwendet werden. Zu Beginn der Analyse sollte aber nur der Standard-Leistungsmonitor verwendet werden. Im Beispiel der Abbildung 17 scheint der Bericht

Auftragsbestätigung an dritter Stelle auf. Die hohe Anzahl an ODBC-Zugriffen erklärt sich aus der oben erwähnten Tatsache, dass ein Bericht aus unzähligen ODBC-Aufrufen bestehen kann, vor allem weil gerade beim Bericht *Auftragsbestätigung* viele Unterberichte verwendet werden. Auffällig lang dauernde Berichte können nun über einen neuen Datenbank-Leistungsmonitor vom Typ DatabaseMonitor-DatabaseAnlaysis oder DatabaseMonitor-FullAnalysis analysiert werden um die zugehörigen Datenbankstatements zu finden. Der neue Datenbank-Leistungsmonitor sollte auf die auffälligen Berichte eingeschränkt werden.

Diese Anwendung hat in der aktuellen Version von Semiramis einen Fehler, da teilweise in der Ausgabe der Name des Berichtes fehlt. Auch irritiert es, dass für einen Bericht die Anzahl an ODBC-Aufrufen angegeben wird. Besser wäre hier, die Anzahl der Ausführungen eines Berichtes anzugeben. Die Anzahl der ODBC-Aufrufe kann als Zusatzinformation dienen. Siehe dazu auch Abschnitt 4.3.

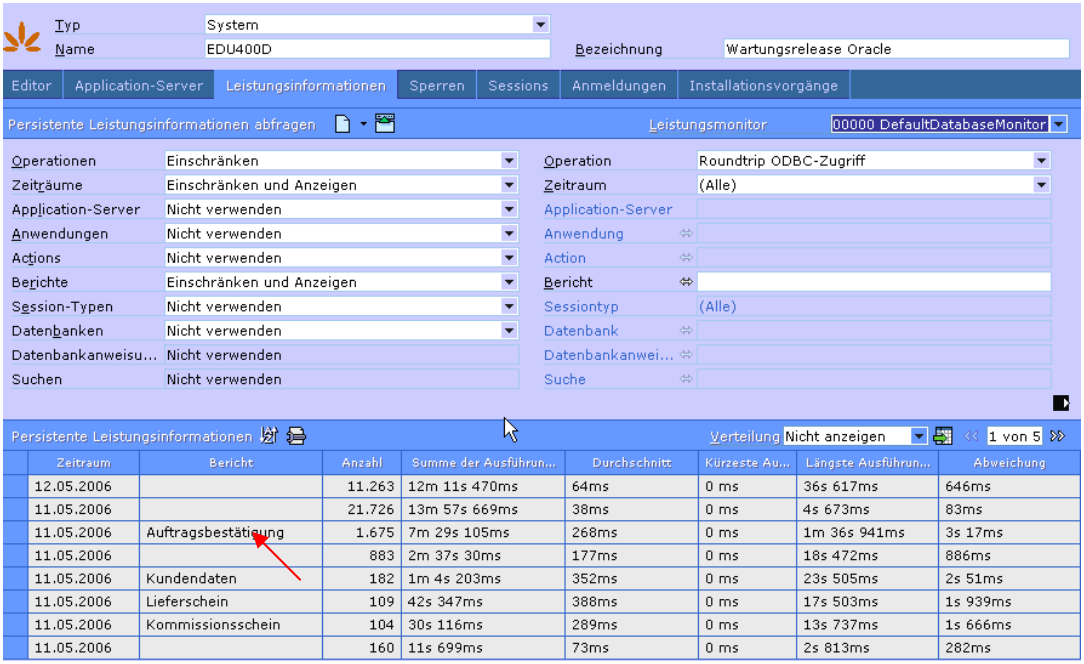


Abbildung 17 – *Zeitaufwändige Berichte sortiert nach Summe* mit dem Standard-Leistungsmonitor.

- Filter *Zeitaufwändige Datenbankanweisungen sortiert nach Summe*: Hier werden Aktionen vom Typ *Datenbankanweisung ausführen* dargestellt und nach Summe ihrer Ausführungszeit sortiert. Diese Anzeige bringt nur für Datenbank-Leistungsmonitore vom Typ DatabaseMonitor-DatabaseAnlaysis oder DatabaseMonitor-FullAnalysis sinnvolle Ergebnisse. Deshalb stellt diese Anzeige den zweiten Schritt der Analyse dar. Zuerst werden mit dem Standard-Leistungsmonitor auffällige Aktionen und Berichte gesucht, bevor ein detaillierterer Datenbank-Leistungsmonitor mit Einschränkung auf die gefundenen Aktionen oder Berichte erstellt wird. Mit Hilfe dieses Filters werden nun die in Summe am längsten dauernden Datenbankanweisungen angezeigt, siehe Abbildung 18.

  Mit Klick auf die Datenbankanweisungen, die an den höchsten Stellen der Liste vorkommen, kommt man in die Anwendung *Datenbankanweisungen abfragen* und

bekommt das zugehörige Statement aufgelistet. Handelt es sich um einen Bericht, muss das Statement dem Entwickler des Berichtes zur Optimierung übergeben werden. Handelt es sich bei einer auffällig zeitaufwändigen Datenbankanweisung um eine Modifikation in Semiramis, so ist das Statement dem Entwickler zur Optimierung zu übergeben. Ist keine Verbesserung am Design des Reports oder der Modifikation möglich, so muss das Statement mit DBMS-Tools gefunden und versucht optimiert zu werden.

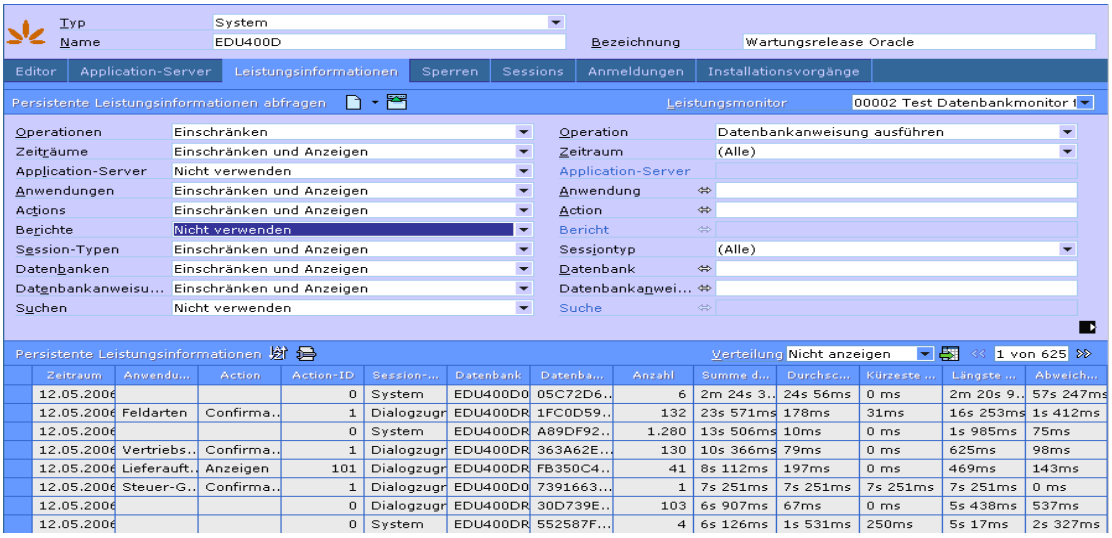


Abbildung 18 – *Zeitaufwändige Datenbankanweisungen sortiert nach Summe* mit dem DatabaseMonitor-DatabaseAnlaysis

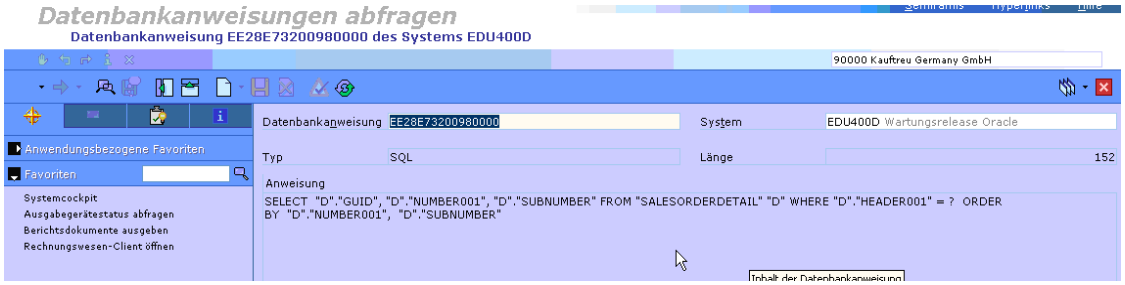


Abbildung 19 – Anwendung *Datenbankanweisungen abfragen*.

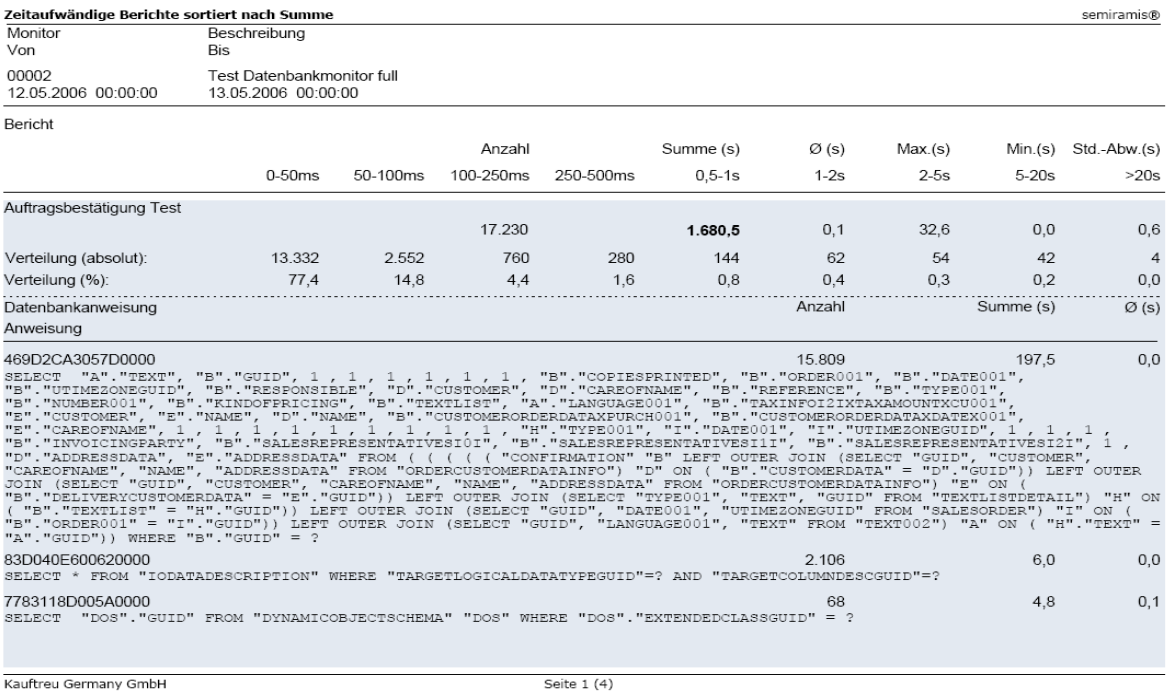
Zusätzlich gibt es für jeden dieser oben genannten Filter auch Filter, die die Antwortzeiten nach Durchschnitt sortieren. Ob nach Durchschnitt oder nach Summe der Ausführungszeit sortiert wird, hängt von der Anwendung bzw. dem Filter ab. Auf Ebene der Datenbankanweisungen empfiehlt es sich, nach Summe der Ausführungszeit zu sortieren, da sich auch die Summe der Anweisungen auf die Gesamtperformance auswirkt. Für Anwendungen und Berichte muss spezifischer vorgegangen werden. Anwendungen und Berichte, die oft ausgeführt werden, sollten aus dem gleichen Grund wie die Datenbankanweisungen nach Summe der Ausführungszeit sortiert werden. Anwendungen und Berichte, die selten ausgeführt werden, aber trotzdem eine zu hohe Ausführungszeit haben, können bei dieser Sortierung übersehen werden. Hohe Ausführungszeiten solcher Anwendungen oder Berichte können von Benutzern aber durchaus als störend empfunden werden. So empfiehlt es sich, eine Sortierung auf durchschnittliche Ausführungszeit anzuwenden, um diese Anwendungen und Berichte zu identifizieren.

Es ist auch möglich, die Verteilung (prozentuell oder absolut) der Antwortzeiten auf Zeitdauerkategorien anzuzeigen. Diese Kategorien sind 0-50ms, 50-100ms, 100-250ms, 250-500ms, 500-1000ms, 1-2s, 2-5s, 5-20s und >20s. In jeder Spalte wird die Summe (oder der prozentuelle Anteil) der Anzahl der in diese Kategorie fallenden Antwortzeiten angezeigt. Mit dieser Ansicht können Ausreißer in der Ausführungszeit identifiziert werden. Liegt beispielsweise eine Anwendung zu 95% in der Kategorie 500-1000ms und zu 5% in der Kategorie >20s, so kann mit hoher Wahrscheinlichkeit davon ausgegangen werden, dass das Problem an anderer Stelle liegt. Eine andere Anwendung hat wahrscheinlich die Performance der betrachteten Anwendung negativ beeinflusst. Diese andere Anwendung sollte daher gesucht werden.

Durch die Verwendung eines Timers, der eine Genauigkeit von 15-20ms hat, kann es vorkommen, dass in der Spalte „kürzeste Ausführungszeit“ der Wert 0ms angezeigt wird (CIS, 2006e). In der Regel werden aber Aktionen sehr oft ausgeführt, so bereinigt sich dieser Wert wieder und die Spalten „Summe der Ausführungszeit“ und „Durchschnitt“ zeigen annähernd richtige Werte an. Trotzdem können hier Verfälschungen auftreten. Da meistens Aktionen von hoher Ausführungszeit interessant sind, fällt dies aber nicht so sehr ins Gewicht. Ab Version 4.3 von Semiramis wird ein Timer verwendet, der eine Genauigkeit von einigen Nanosekunden hat (CIS, 2006e).

3.3.1.2 Performance-Berichte

Die oben erwähnten Filter sind auch als Berichte verfügbar. Sie zeigen alle Informationen, die auch im *Systemcockpit* verfügbar sind, sind jedoch übersichtlicher aufbereitet. Die Zeitdauerkategorien werden ebenfalls angezeigt. Falls eine Datenbankanweisung über einen detaillierteren Datenbank-Leistungsmonitor mitgeloggt wurde, wird diese vollständig angezeigt. Diese Berichte sind mit Crystal Reports erstellt, und können somit nach Wunsch angepasst werden.



Zeitaufwändige Berichte sortiert nach Summe semiramis®

Monitor	Beschreibung
Von	Bis
00002	Test Datenbankmonitor full
12.05.2006 00:00:00	13.05.2006 00:00:00

Bericht

	Anzahl				Summe (s)	Ø (s)	Max.(s)	Min.(s)	Std.-Abw.(s)
	0-50ms	50-100ms	100-250ms	250-500ms	0,5-1s	1-2s	2-5s	5-20s	>20s
Auftragsbestätigung Test									
			17.230		**1.680,5**	0,1	32,6	0,0	0,6
Verteilung (absolut):	13.332	2.552	760	280	144	62	54	42	4
Verteilung (%):	77,4	14,8	4,4	1,6	0,8	0,4	0,3	0,2	0,0

Datenbankanweisung Anweisung	Anzahl	Summe (s)	Ø (s)
469D2CA3057D0000	15.809	197,5	0,0

```
SELECT  "A"."TEXT", "B"."GUID", 1 , 1 , 1 , 1 , 1 , 1 , "B"."COPIESPRINTED", "B"."ORDER001", "B"."DATE001",
"B"."UTIMEZONEGUID", "B"."RESPONSIBLE", "D"."CUSTOMER", "D"."CAREOFNAME", "B"."REFERENCE", "B"."TYPE001",
"B"."NUMBER001", "B"."KINDOFPRICING", "B"."TEXTLIST", "A"."LANGUAGE001", "B"."TAXINFOI2IXTAXAMOUNTXCU001",
"E"."CUSTOMER", "E"."NAME", "D"."NAME", "B"."CUSTOMERORDERDATAXPURCH001", "B"."CUSTOMERORDERDATAXDATEX001",
"E"."CAREOFNAME", 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , "H"."TYPE001", "I"."DATE001", "I"."UTIMEZONEGUID", 1 , 1 , 1 ,
"B"."INVOICINGPARTY", "B"."SALESREPRESENTATIVESI0I", "B"."SALESREPRESENTATIVESI1I", "B"."SALESREPRESENTATIVESI2I", 1 ,
"D"."ADDRESSDATA", "E"."ADDRESSDATA" FROM ( ( ( ( "CONFIRMATION" "B" LEFT OUTER JOIN (SELECT "GUID", "CUSTOMER",
"CAREOFNAME", "NAME", "ADDRESSDATA" FROM "ORDERCUSTOMERDATAINFO") "D" ON ( "B"."CUSTOMERDATA" = "D"."GUID")) LEFT OUTER
JOIN (SELECT "GUID", "CUSTOMER", "CAREOFNAME", "NAME", "ADDRESSDATA" FROM "ORDERCUSTOMERDATAINFO") "E" ON (
"B"."DELIVERYCUSTOMERDATA" = "E"."GUID")) LEFT OUTER JOIN (SELECT "TYPE001", "TEXT", "GUID" FROM "TEXTLISTDETAIL") "H" ON
( "B"."TEXTLIST" = "H"."GUID")) LEFT OUTER JOIN (SELECT "GUID", "DATE001", "UTIMEZONEGUID" FROM "SALESORDER") "I" ON (
"B"."ORDER001" = "I"."GUID")) LEFT OUTER JOIN (SELECT "GUID", "LANGUAGE001", "TEXT" FROM "TEXT002") "A" ON ( "H"."TEXT" =
"A"."GUID")) WHERE "B"."GUID" = ?
```

83D040E600620000	2.106	6,0	0,0

```
SELECT * FROM "IODATADESCRIPTION" WHERE "TARGETLOGICALDATATYPEGUID"=? AND "TARGETCOLUMNDESCGUID"=?
```

7783118D005A0000	68	4,8	0,1

```
SELECT  "DOS"."GUID" FROM "DYNAMICOBJECTSCHEMA" "DOS" WHERE "DOS"."EXTENDEDCLASSGUID" = ?
```

Kauftreu Germany GmbH Seite 1 (4)

Abbildung 20 – *Zeitaufwändige Berichte sortiert nach Summe ausgeben* für den DatabaseMonitor-FullAnalysis.

Zusätzlich ist auch ein Bericht *Cache Statistikeinträge ausgeben* verfügbar. Alle diese Berichte können auf einen Application-Server, einen Zeitraum, und mit Ausnahme des Berichtes *Cache Statistikeinträge ausgeben* auf die Anwendung, den Leistungsmonitor und eine maximale Anzahl von Ergebnissen eingeschränkt werden. Die Namen der Berichte lauten:

- *Cache-Statistikeinträge ausgeben*: Dieser Bericht gibt Statistiken der einzelnen Business Objects aus. Es sind dies Anzahl der Cache Reads, Database Reads, Inserts, Updates und Deletes.
- *Zeitaufwändige Aktionen sortiert nach Summe ausgeben*: Dieser Bericht filtert nach Aktionen vom Typ *Roundtrip performAction*. Die Anzeige ist ähnlich dem gleichnamigen Filter im *Systemcockpit*.
- *Zeitaufwändige Berichte sortiert nach Summe ausgeben*: Dieser Bericht filtert nach Aktionen vom Typ *Roundtrip ODBC-Zugriff*. Die Anzeige ist ähnlich dem gleichnamigen Filter im *Systemcockpit*.
- *Zeitaufwändige Datenbankanweisungen sortiert nach Summe ausgeben*: Dieser Bericht filtert nach Aktionen vom Typ *Datenbankanweisung ausführen*. Die Anzeige ist ähnlich dem gleichnamigen Filter im *Systemcockpit*. Hier bringt wie im *Systemcockpit* die Verwendung des Standard-Leistungsmonitors keine aussagekräftigen Ergebnisse.

Abbildung 20 zeigt ein Beispiel für den Bericht *Zeitaufwändige Berichte sortiert nach Summe ausgeben* für den DatabaseMonitor-FullAnalysis. Der manipulierte Bericht *Auftragsbestätigung* taucht an erster Stelle auf. Die Fehler vom *Systemcockpit* treten hier nicht auf. Es werden hier auch die Datenbank-Statements der Unterberichte angezeigt (in der Abbildung ist nur die erste von vier Seiten dargestellt). Auch im Bericht *Zeitaufwändige Datenbankanweisungen sortiert nach Summe ausgeben* kommt das erste Statement (469D2CA3057D0000) an zweiter Stelle vor. Dieser Bericht ist hier nicht dargestellt. Es sollte also nun mit dem Entwickler des Berichtes *Auftragsbestätigung* gesprochen werden, ob dieser optimiert werden kann. Das Statement sollte auch dem Entwickler übergeben werden.

3.3.1.3 Datei-Leistungsmonitoring

Um noch detailliertere Analysen zu erhalten, kann das Datei-Leistungsmonitoring verwendet werden. Auch aufgrund der in der aktuellen Semiramis Version enthaltenen Fehler empfiehlt es sich – bis die Fehler behoben sind – für Detailanalysen das bereits länger implementierte und zuverlässiger arbeitende Datei-Leistungsmonitoring zu verwenden. Auch für das Datei-Leistungsmonitoring müssen in der Anwendung *Leistungsmonitore* Monitore aktiviert werden. Diese speichern aber im Unterschied zu den Datenbank-Leistungsmonitoren Informationen in Logfiles, die durch die Software *Sawmill* ausgewertet werden können. Es gibt unterschiedliche Arten von Monitoren. Die Monitore für die Analyse im Entwicklungssystem belasten das System stark, dagegen ist die Belastung der Monitore für die Analyse von Produktivsystemen nicht so hoch, da sie weniger Parameter aufzeichnen, und die Schwellwerte niedriger sind. Deshalb werden nur die Monitore für Produktivsysteme ausführlicher behandelt. Für Details und Erklärungen zu

den einzelnen Operationen siehe (CIS, 2006c). Für die Konfiguration des Leistungsmonitors stehen folgende Vorlagen zur Verfügung:

- Datei-Leistungsmonitor zur vollständigen Analyse im Entwicklungssystem. Es werden alle verfügbaren Operationen ohne Schwellwert aufgezeichnet.
- Datei-Leistungsmonitor zur Analyse im Entwicklungssystem. Es werden alle verfügbaren Operationen mit Schwellwerten aufgezeichnet.
- Datei-Leistungsmonitor zur Analyse im Produktivsystem. Es werden nur die Operationen *DB_STATEMENT_EXECUTE* (Datenbankanweisung ausführen) mit Schwellwert 0,5s, *DB_TRANSACTION_COMMIT (*erfolgreiches Beenden einer Datenbanktransaktion) mit Schwellwert 1s, *ROUNDTRIP_PERFORM_ACTION* (Antwortzeit einer Anwendung) mit Schwellwert 1s, *ROUNDTRIP_GUI* (Antwortzeit einer Anwendung, inkl. Eingabe und Darstellung auf dem Browser des Benutzers) mit Schwellwert 1s, *ROUNDTRIP_ODBC* (Aufruf einer ODBC-Funktion während der Berichtsausführung) mit Schwellwert 1s und *LOCK_AQUIRE_WAIT* (Wartezeit, bis eine Sperre gesetzt werden kann) mit Schwellwert 10 s protokolliert.
- Datei-Leistungsmonitor zur Trendanalyse im Produktivsystem. Es werden nur die Operationen *DB_STATEMENT_EXECUTE* mit Schwellwert 3s, *DB_TRANSACTION_COMMIT* mit Schwellwert 5s, *ROUNDTRIP_PERFORM_ACTION* mit Schwellwert 5s, *ROUNDTRIP_GUI* mit Schwellwert 5s und *ROUNDTRIP_ODBC* mit Schwellwert 5s protokolliert.

Diese Vorlagen sind Standardvorlagen, die mit Semiramis ausgeliefert werden. Die Schwellwerte sind von der Cross Industrie Software (C.I.S.) AG aus Erfahrungswerten so gewählt worden (CIS, 2006c). Die Vorlagen sind Extensible Markup Language (XML)-Dateien, und können so nach Wunsch angepasst werden.

Die von den Datei-Leistungsmonitoren erzeugten Dateien müssen nun aus Performancegründen auf einen anderen Rechner transportiert werden und mit der Software *Sawmill* ausgewertet werden. In der Standard Auslieferung von Semiramis ist eine Konfigurationsdatei für *Sawmill* mitgeliefert. *Sawmill* bietet für jede Kontextinformation (Benutzer, Business Object, Datenbankanweisung, usw.) eine eigene Ansicht, die am häufigsten auftretende Werte für jede gewählte Kontextinformation anzeigt. *Sawmill* zählt das Auftreten jedes gefundenen Wertes und listet die Werte in absteigender Reihenfolge mit ihrer gefundenen Anzahl auf. Zusätzlich wird die summierte Ausführungsdauer angezeigt. *Sawmill* unterscheidet sich hier also von der Anzeige im *Systemcockpit*, wo nach der Gesamtsumme der Ausführungsdauer sortiert wird. Durch das Festlegen von Filtern können die folgenden Abfragen auf einen oder mehrere konkrete Werte für eine oder mehrere Kontextinformationen eingeschränkt werden. Es werden dann nur noch die Operationen betrachtet, die die Filterkriterien erfüllen. Somit kann man ein Problem immer weiter eingrenzen. Dies ist im *Systemcockpit* nicht möglich. Für Beispiele siehe (CIS, 2006c, S. 28f).

Der Startpunkt der Analyse mit *Sawmill* ist die Ansicht *Zeitdauerkategorie*. Diese Ansicht zeigt die häufigsten aufgetretenen Ausführungszeitkategorien (die Kategorien sind gleich denen in den Performance-Berichten gewählt) aggregiert über die protokollierten Operationen an. Es kann nun

die gewünschte Kategorie gewählt werden, beispielsweise > 20.000 ms. Nun kann man sich über die weiteren Filter die Operationen anzeigen lassen, die in diese Zeitdauerkategorie fallen. Zu jeder der nun aufgelisteten Operationen kann man sich die zugehörige Datenbank-Statement-ID (zum Anzeigen des kompletten Statements muss die Anwendung *Datenbankabfragen anzeigen* ausgeführt werden, und die ID kopiert werden) anzeigen lassen. Weiters kann auch die zugehörige Anwendung oder der zugehörige Bericht angezeigt werden.

3.3.1.4 Weitere Leistungsinformationen im *Systemcockpit*

Das *Systemcockpit* bietet in weiteren Karteireitern noch weitere Leistungsinformationen, nachfolgend die wichtigsten (CIS, 2006a; CIS, 2006b):

- *Sessions*: Hier werden alle aktiven Sessions auf dem System samt ihrem Anteil an der CPU-Zeit angezeigt. Zugehörige Threads werden im Typ *Application-Server* angezeigt. Sessions können auch manuell beendet werden.
- *Threads* (nur im Typ *Application-Server* verfügbar): Hier werden alle aktiven Threads auf einem SAS samt ihrem Anteil an der CPU-Zeit und ihrer Priorität (Semiramis verwaltet interne Prioritäten für Threads, 1 ist die niedrigste, 10 die höchste. So kann z.B. ein Benutzer eine Priorität für einen Hintergrundauftrag vergeben) angezeigt.
- *Persistenzdienst* (nur im Typ *Application-Server* verfügbar): Hier werden die Anzahl der Zugriffe auf den Persistenzdienst gruppiert nach Cache-Strategie (LRU, alle Instanzen, Permanent) aufgelistet. Zudem wird die Semiramis Buffer Hit-Ratio angezeigt. Diese sollte so nahe wie möglich bei 100% liegen.
- *Sperren* (nur im Typ *System* verfügbar): Hier werden Sperren innerhalb des Systems angezeigt. Dies sind eigene Sperren von Semiramis auf Business Objects, die zusätzlich zu Datenbanksperren gehalten werden. Sperren sollten so selten wie möglich auftreten.
- *Application-Server*, *Schaltfläche Speicherverbrauch* (nur im Typ *Application-Server* verfügbar): Hier kann der verwendete, der verfügbare und der maximale Heap-Speicher angezeigt werden.
- *Zugriffsstatistik* (nur im Typ *Application-Server* verfügbar): Hier werden die Anzahl der Zugriffe, die Zugriffsstrategie (LRU, alle Instanzen, Permanent) und die Cache Hit-Ratio in Prozent pro Business Object angezeigt.

3.3.1.5 Zusammenfassung

Für eine Zusammenfassung und eine kritische Betrachtung des Semiramis-Leistungsmonitoring siehe Abschnitt 4.3. Abbildung 21 zeigt eine Zusammenfassung der Vorgehensweise zur Identifizierung von Anwendungen und Berichten mit hoher Ausführungszeit und deren zugehörigen Datenbankanweisungen mit dem Semiramis-Leistungsmonitoring. Sie wird im Weiteren zur Erstellung der Dokumentenvorlage verwendet.

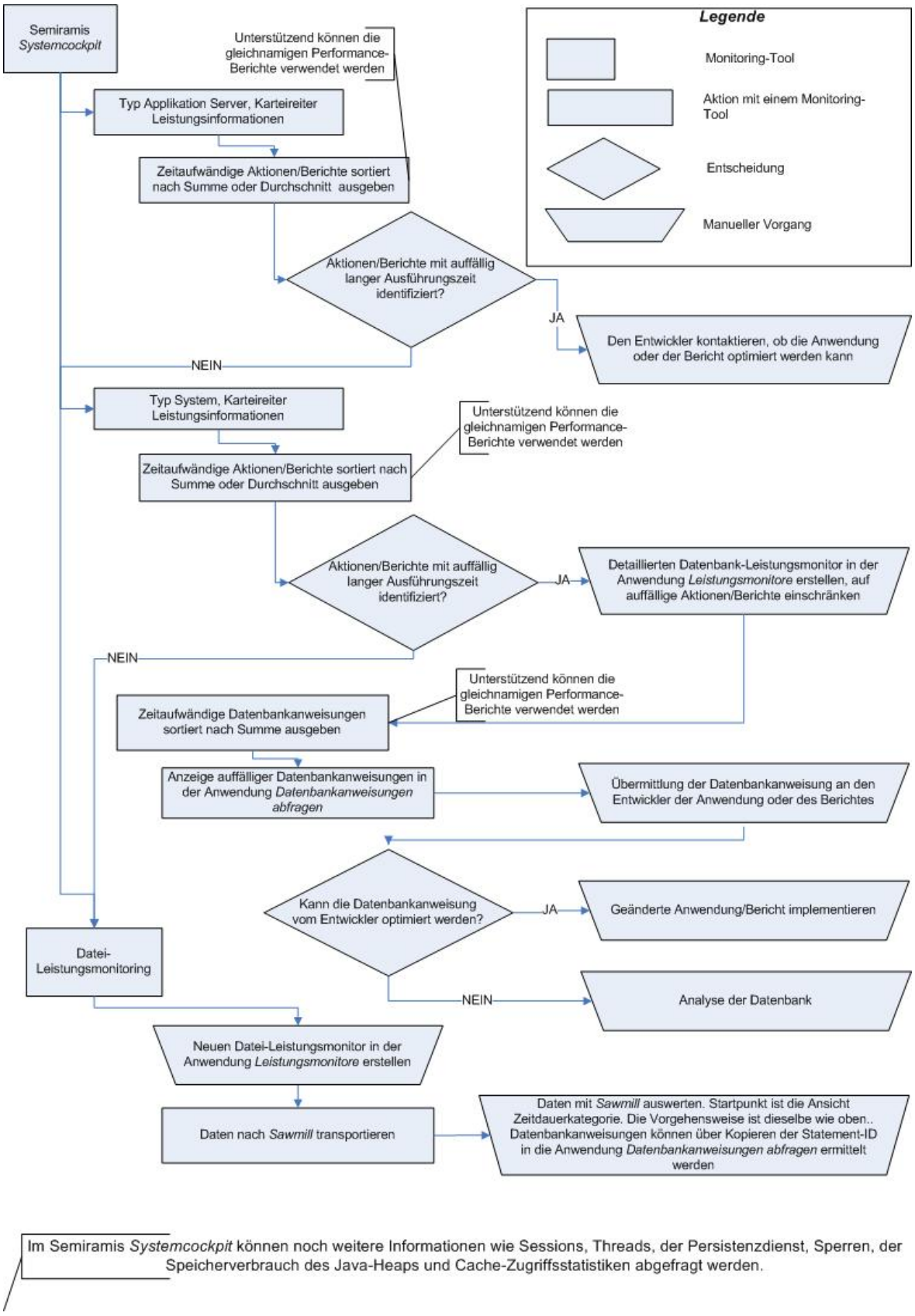


Abbildung 21 – Zusammenfassung des Semiramis-Leistungsmonitorings.

3.3.2 OpenNMS

OpenNMS ist ein von einem Diplomanden entwickeltes web-basiertes Programm, welches über das Simple Network Management Protokoll (SNMP) und die Java Management Extension (JMX) Informationen über Semiramis und der JVM pro SAS anzeigen kann (Wurzrainer, 2006). Auch Betriebssystemdaten werden angezeigt. Diese Daten werden grafisch aufbereitet. Historische Daten werden ebenfalls zur Verfügung gestellt. Derzeit verfügbare Kennzahlen sind:

- CPU-Auslastung des Rechners auf dem der überwachte SAS läuft.
- Cache Hit-Ratio des Semiramis Cache.
- HTTPS Response Time des SAS.
- Verfügbarer Arbeitsspeicher des Rechners auf dem der überwachte SAS läuft.
- JVM Heap Speicher.
- Eden, Tenured und Survivor Space der JVM.
- Permanent Space (hier als Non Heap Memory bezeichnet) der JVM.
- Anzahl der Garbage Collections.

OpenNMS könnte auf alle weiter unten beschriebenen Kennzahlen der DBMS und Betriebssysteme erweitert werden. Auch die Informationen des Semiramis-Leistungsmonitorings könnten über JMX ausgewertet werden.

3.3.3 JVM

Die Analyse der JVM kann mit dem in der Version 1.5 gelieferten Programm *jconsole* durchgeführt werden (Chung, 2004).

Die *jconsole* besteht aus sechs Karteireitern:

- *Summary*: Zeigt zusammenfassende Informationen wie die Laufzeit, CPU-Zeit, Anzahl der Threads, Größe des Heaps, zugesagter Speicher, Garbage Collection Einstellungen, geladene Klassen, sowie Speicherdaten vom Betriebssystem an. Siehe dazu Abbildung 22.
- *Memory*: Zeigt Informationen über die einzelnen Speicherbereiche, Eden, Tenured, Survivor, sowie Permanent Space an. Auch Informationen über die Garbage Collection werden angezeigt. Diese kann manuell ausgelöst werden.
- *Threads*: Zeigt Informationen über Threads.
- *Classes*: Zeigt Informationen über geladene Klassen.
- *MBeans*: Zeigt Informationen über geladene MBeans.

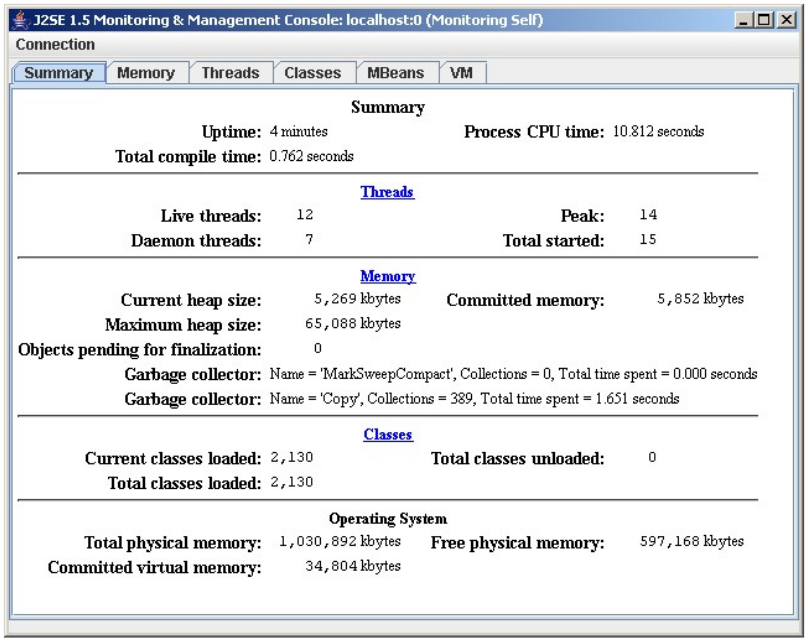


Abbildung 22 – *jconsole*, Reiter Summary (Quelle: Chung, 2004).

Eine Auflistung aller JVM Tuning-Parameter für Semiramis findet sich im Dokument „JVM-Einstellungen“ (CIS, 2006d). Diese Parameter unterscheiden sich je nach Implementierung (IBM, SUN, 32 oder 64 Bit) sehr stark. Auch die Performance der unterschiedlichen Implementierungen kann sehr unterschiedlich sein. Für Details zur Analyse der JVM siehe (Wurzrainer, 2006).

3.3.4 Datenbanken

Ist keine Verbesserung am Design von SQL-Statements von Berichten oder Modifikationen möglich, oder hat die Optimierung der Statements keine Verbesserung gebracht, folgt die Analyse der Datenbanken. Diese beginnt mit der Analyse der Datenbankbuffer und der Identifizierung von teueren SQL-Anweisungen. Es folgen die Identifizierung von Schreib/Lese- bzw. Sperrproblemen sowie sonstige Tuning-Maßnahmen und Monitoring-Möglichkeiten.

3.3.4.1 Analyse der Datenbank-Buffer

Die wichtigste Kennzahl für die Analyse der Datenbank-Buffer ist die Buffer Hit-Ratio. Diese kann durch Beseitigung von teuren SQL-Anweisungen (siehe weiter unten) oder durch Vergrößern der Datenbankbuffer erhöht werden. Das Ziel ist es – bei einem „warmen“ System, d.h. das System läuft schon eine geraume Zeit, und fast alle oft benötigten Daten sind schon im Buffer – die Buffer Hit-Ratio nahe 100% zu bringen. Die Verdrängungsrate (Seitenersetzung) sollte gegen 0 gehen. Es sollte ausreichend Speicherplatz im Buffer zur Verfügung stehen (Schneider, 2005).

Oracle

Eine Analyse der Datenbank-Buffer für Oracle wird am Besten über einen *Snapshot-Report* durchgeführt. Dieser Report ist auf der *Performance* Seite des *Oracle Enterprise Managers*

aufrufbar, siehe dazu auch Abschnitt 3.3.4.5. Dabei sind die folgenden Kennzahlen von Bedeutung (Choi, 2005; Chan, 2005):

- Hit-Ratios (finden sich im Bereich *Instance Efficiency Percentages*): „Buffer Cache Hit Ratio“ und „Shared Pool Hit Ratio“ sollten wie bereits erwähnt so nahe wie möglich bei 100% liegen. Auch die Werte „Soft Parse %“ und „Latch Hit %“ sollten so hoch wie möglich sein.
- „Shared Pool Memory Usage %“ (findet sich im Bereich *Shared Pool Statistics*): Es sollte genug Speicher zur Verfügung stehen. Je nach Anwendungsfall kann hier ein unterschiedlicher Prozentsatz sinnvoll sein.
- „Buffer Gets“, „Physical Reads“, „Physical Writes“ (finden sich im Bereich *Buffer Pool Statistics*): Wenn diese Werte konstant hoch, und die Hit-Ratios konstant niedrig sind, signalisiert dies ein Speicherproblem.

SQL-Server

Die SQL-Server Kennzahlen können über den *perfmon* im Objekt *SQLServer:Puffer-Manager* angezeigt werden. Die wichtigsten Kennzahlen für die Analyse der Datenbank-Buffer sind (Schwartz, 2005):

- „Puffercache-Trefferquote“: Sollte wie bei Oracle knapp bei 100% liegen.
- „Freie Seiten“: Es sollten genügend freie Seiten vorhanden sein. Wie bei Oracle kann hier keine generelle Empfehlung abgegeben werden.
- „Anhalten der Freiliste/Sekunde“: Anzahl der Anforderungen, die auf eine freie Seite warten mussten. Ein Wert von 3 oder 4 signalisiert ein Problem.
- „Gestohlene Seiten“: Anzahl der Seiten, die für sonstige Serverzwecke verwendet werden, d.h. der Datenbank vom Betriebssystem „gestohlen“ wurden. Ein hoher Wert signalisiert, dass zu wenig Hauptspeicher im System vorhanden ist, oder zu viele andere Applikationen auf dem System laufen.
- „Seitenlesevorgänge/Sekunde“, „Seitenschreibvorgänge/Sekunde“: Auch hier gilt, wenn beide Werte konstant hoch sind, und die „Puffercache-Trefferquote“ konstant niedrig ist, signalisiert dies ein Speicherproblem.

DB2 UDB for iSeries

Da die DB2 UDB for iSeries fest in das Betriebssystem der iSeries integriert ist, gibt es kein zusätzliches Tool zur Abfrage dieser Kennzahlen. Die Datenbank läuft in einem eigenen Subsystem mit eigenem Speicherpool. Für die Analyse der Kennzahlen ist der Speicherpool der Datenbank auszuwählen. Die folgenden Kennzahlen können über eine *Systemüberwachung* im *Management Central* des *iSeries Navigator*, über den *Systemstatus* des *iSeries Navigator*, bzw. über den Befehl *wrksyssts* abgefragt werden (IBM, 2005; KTW, 2006). Details dazu finden sich in Abschnitt 3.3.5.3.

- „Logische Datenbank-E/A im Stapelbetrieb“ (*Systemüberwachung* im *Management Central*): Diese Kennzahl gibt die durchschnittliche Anzahl logischer Ein- bzw. Ausgabeoperationen in der Datenbank an, die pro Sekunde auf dem System ausgeführt werden.
- Buffer-Trefferquoten: Es gibt keine eigene Anzeige von Buffer-Trefferquoten. Mit dem Befehl *wrksyssts* können aber die Seitenfehler pro Speicherpool für Datenbank- und nicht-Datenbankzugriffe angezeigt werden. Ist der Wert für die Datenbank-Seitenfehler über einen längeren Zeitraum hoch, lässt dies auf eine schlechte Buffer-Trefferquote schließen.
- Speicherverwendung: Analyse des Speicherpools der Datenbank über den *Systemstatus*, siehe Abschnitt 3.3.5.3.

3.3.4.2 Identifizierung teurer SQL-Anweisungen

Die Identfizierung von teuren SQL-Anweisungen läuft auf allen drei Datenbank-Plattformen nach dem gleichen Schema:

- Zuerst müssen die SQL-Statements gesucht werden, die eine hohe Ausführungszeit bzw. einen hohen Anteil an der Gesamtaktivität des Systems haben. Oracle gibt den prozentuellen Anteil der Top 10 SQL-Anweisungen an der Gesamtaktivität des Systems an und sortiert diese gleich in absteigender Reihenfolge. Beim SQL-Server und bei der DB2 UDB for iSeries wird die Ausführungszeit pro Statement angegeben. Um Statements mit hoher Ausführungszeit zu finden, muss nach der Ausführungszeit in absteigender Reihenfolge sortiert werden. Zu beachten ist, dass im Unterschied zum Semiramis-Leistungsmonitoring hier jedes einzelne Statement aufgelistet ist, und keine Aggregation stattfindet. Um Aggregationen durchzuführen, müssen die Daten in einer Tabelle gespeichert, und per SQL ausgewertet werden. So lassen sich die Anzahl der Ausführungen, die summierte Ausführungszeit, oder die durchschnittliche Ausführungszeit eines Statements anzeigen. Meistens wurde jedoch das zu optimierende Statement bereits durch das Semiramis-Leistungsmonitoring gefunden, so muss nur noch dieses Statement gefunden werden. Dieses Statement zu finden, ist aber oft nicht so einfach, da sich die Statement-ID’s und auch die Syntax der Statements zwischen Semiramis und dem DBMS unterscheiden. Die beste Möglichkeit ein Statement zu suchen, das im Semiramis-Leistungsmonitoring gefunden wurde, ist es, die Anwendung welches das Statement verwendet (wird im Leistungsmonitor angegeben) erneut auszuführen. Lässt man nun gleichzeitig eine Ablaufverfolgung auf der Datenbank laufen, sollte dieses Statement rasch gefunden werden.
- Analyse des Zugriffsplans und Optimieren des Statements beispielsweise durch Indizes.
- Testen, ob die Optimierung eine Verbesserung gebracht hat. Im Fall eines Index, Überprüfung ob dieser verwendet wird. Die Ausführungszeit des optimierten Statements sollte mit der des alten Statements verglichen werden. Der aussagekräftigste Test ist es aber, das System mit realen Benutzern zu testen. Eine Optimierung hat nur dann etwas gebracht, wenn dies von einem Benutzer spürbar wahrgenommen wird.

- Wenn der Index eine Verbesserung gebracht hat, muss dieser in Semiramis in der Anwendung *Individuelle Indizes* angelegt werden, da er ansonsten bei der nächsten Software-Aktualisierung nicht mehr vorhanden ist.

Nachfolgend ein ausführliches Beispiel für die Oracle Datenbank, die zwei weiteren DBMS werden nur grob behandelt, da die Vorgehensweise ähnlich ist.

Oracle

Der *Oracle Enterprise Manager* der Version 10g bietet eine *Database Performance Page*, die einen groben Überblick über den Zustand der Datenbank gibt (Choi, 2005; Chan, 2005). Über den Link *Top Aktivität* kommt man auf eine Seite, die unter Anderem die zehn teuersten SQL-Anweisungen anzeigt, siehe Abbildung 23.

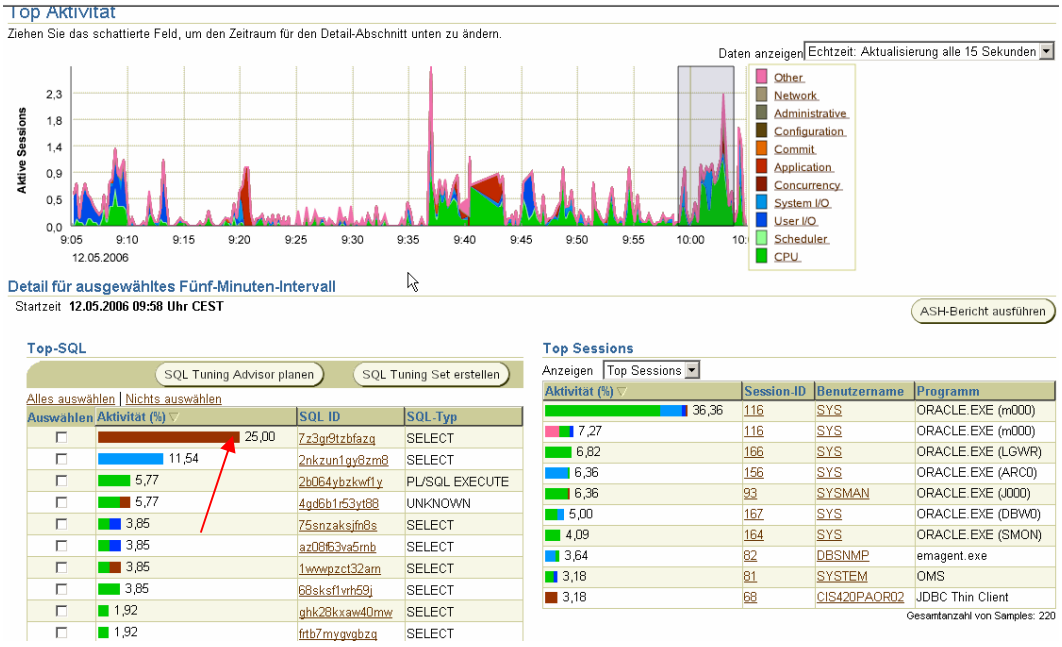


Abbildung 23 – *Oracle Enterprise Manager*, Link *Top Aktivität*.

In der Abbildung 23 im Abschnitt *Top-SQL* fällt gleich das erste Statement auf, das einen Anteil von 25% an der Gesamtaktivität des Systems hat. Durch einen Klick auf das Statement kommt man in die *SQL-Details Page* und das Statement wird angezeigt. Wie in Abbildung 24 zu sehen ist, handelt es sich um dasselbe Statement, das auch schon mit dem Semiramis-Leistungsmonitoring – mit dem Bericht *Zeitaufwändige Berichte sortiert nach Summe ausgeben* – gefunden wurde (siehe Abbildung 20).

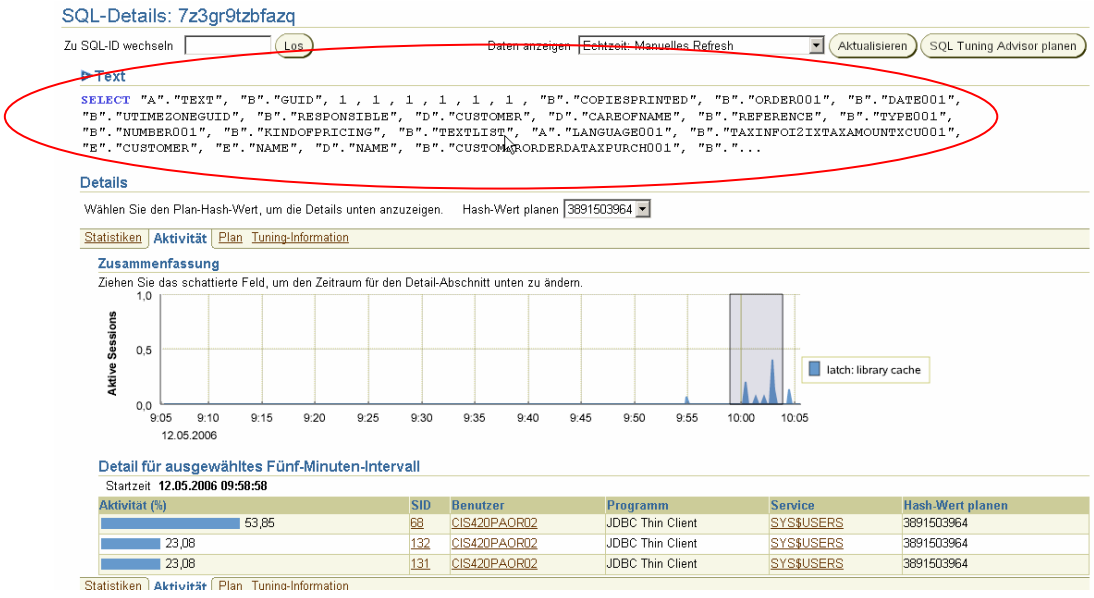


Abbildung 24 – *Oracle Enterprise Manager*, Link *SQL Details*.

Vorgang	Objekt	Objekttyp	Reihenfolge	Zeilen	Größe (KB)	Kostenfaktor	Zeit (s)	CPU-Kostenfaktor	I/O-Kosten
SELECT STATEMENT			18			8			
NESTED LOOPS OUTER			17	4	2,297	8	1	72272	8
NESTED LOOPS OUTER			14	1	0,538	5	1	48227	5
NESTED LOOPS OUTER			11	1	0,502	4	1	40906	4
NESTED LOOPS OUTER			8	1	0,385	3	1	32424	3
NESTED LOOPS OUTER			5	1	0,268	2	1	23943	2
TABLE ACCESS BY INDEX ROWID	CONFIRMATION	TABLE	2	1	0,227	1	1	14021	1
INDEX UNIQUE SCAN	SYS_C00345948	INDEX (UNIQUE)	1	1		0		1050	0
TABLE ACCESS BY INDEX ROWID	SALESORDER	TABLE	4	224	9,188	1	1	9921	1
INDEX UNIQUE SCAN	SYS_C00351855	INDEX (UNIQUE)	3	1		0		1050	0
TABLE ACCESS BY INDEX ROWID	ORDERCUSTOMERDATAINFO	TABLE	7	36	4,219	1	1	8481	1
INDEX UNIQUE SCAN	SYS_C00349960	INDEX (UNIQUE)	6	1		0		1050	0
TABLE ACCESS BY INDEX ROWID	ORDERCUSTOMERDATAINFO	TABLE	10	36	4,219	1	1	8481	1
INDEX UNIQUE SCAN	SYS_C00349960	INDEX (UNIQUE)	9	1		0		1050	0
TABLE ACCESS BY INDEX ROWID	TEXTLISTDETAIL	TABLE	13	1	0,036	1	1	7321	1
INDEX RANGE SCAN	SYS_C00352937	INDEX (UNIQUE)	12	1		1	1	7321	1
TABLE ACCESS BY INDEX ROWID	TEXT002	TABLE	16	4	0,145	3	1	24044	3
INDEX RANGE SCAN	SYS_C00352946	INDEX (UNIQUE)	15	4		1	1	7921	1

Abbildung 25 – *Oracle Enterprise Manager*, Reiter *Plan*.

Über den Reiter *Plan* kann der Ausführungsplan angezeigt werden, siehe Abbildung 25.

Mit dem *Tuning Advisor* können nun Empfehlungen angezeigt werden. Wie Abbildung 26 zeigt, sollten für das Statement neue Indizes erstellt werden. Diese zwei Indizes sollen den Index Range Scan in einen Index Uniqe Scan umwandeln. Das heisst, es werden Indizes auf die *WHERE* Bedingungen erstellt, und so bessere Performance erzielt. Diese Indizes sollen nun implementiert werden. Wenn der nachfolgende Test Verbesserungen gebracht hat, dann müssen die Indizes in Semiramis angelegt werden. Es muss auch kontrolliert werden, ob die Indizes tatsächlich verwendet werden. Die einfachste Variante ist hier, erneut den Ausführungsplan anzuzeigen und dort zu kontrollieren, ob der Index verwendet wird.

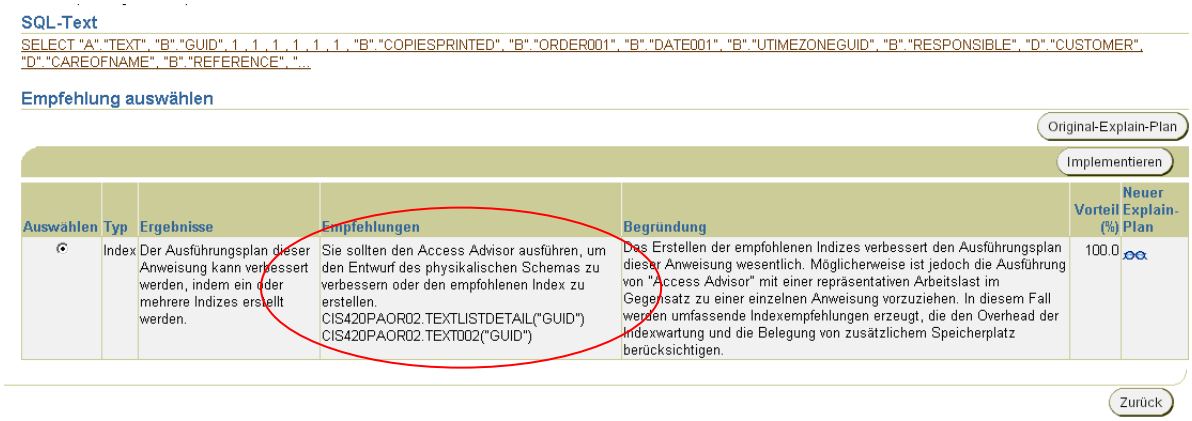


Abbildung 26 – *Oracle Enterprise Manager*, Link *Tuning Advisor*.

SQL-Server

SQL-Server 2005 bietet drei Tools an: das *SQL-Server Management Studio*, das auch den *Query Analyzer* beinhaltet, den *SQL-Server Profiler* und den *Database Tuning Advisor* (*DTA*). Details dazu finden sich in (Jones, 2002). Um teure SQL-Anweisungen zu finden muss im *SQL-Server Profiler* eine neue *Ablaufverfolgung* erstellt werden. Es reicht aus, den Start und das Ende der ausgeführten Transaktionen mitzuverfolgen. Deshalb werden in der neuen *Ablaufverfolgung* nur die Ereignisse RPC:Starting und RPC:Completed ausgewählt. Der Besonderheit des SQL-Servers, der Autoparametrisierung, muss hier spezielle Aufmerksamkeit geschenkt werden. In der *Ablaufverfolgung* werden zwei Arten von Statements angezeigt. Ein Statement, das mit *declare* beginnt, erzeugt die *stored procedure* und beinhaltet die SQL-Anweisung. Das Statement, welches mit *execute* beginnt, führt die *stored procedure* aus, siehe dazu Abbildung 27 und Abbildung 28. Die Vorgangsweise ist also folgende: Zuerst müssen alle Statements, die mit *execute* beginnen, nach Ausführungszeit sortiert werden. Hat man die teuren Anweisungen identifiziert, muss anhand der Zahl nach dem *sp_execute* Statement (im Beispiel die Zahl 102) das zugehörige *declare* Statement gesucht werden (erkennbar durch die Zeile *set @p1=102*). Da der *SQL-Server Profiler* Ergebnisse nicht sortieren kann, empfiehlt es sich, die Ergebnisse des *SQL-Server Profilers* in eine Tabelle zu speichern, und diese dann per SQL auszuwerten. Hat man das Statement gefunden, muss dieses zusammengesetzt werden (nur das SQL-Statement aus der *declare* Anweisung, den Parameter *@p1* mit der Zahl nach dem *sp_execute* ersetzen. Im Beispiel also *SELECT A.TEXT, B.GUID, … WHERE B.GUID = '0x0060370B33682110BEDFC0A801060000')*, und der zugehörige Ausführungsplan mit der Aktion *geschätzten Ausführungsplan anzeigen* im *Query Analyzer* angezeigt werden. Mit dem *DTA* kann das Statement optimiert werden Dieser Vorgang wurde wieder mit dem Bericht *Auftragsbestätigung* durchgeführt. Der SQL-Server hat kein Optimierungspotential gefunden, da im Ausführungsplan nur Clustered Index-Suchen vorgekommen sind (ohne Abbildung). Die Verwendung von Indizes kann im *DTA* durch den Bericht *Indexverwendungsbericht* überprüft werden.

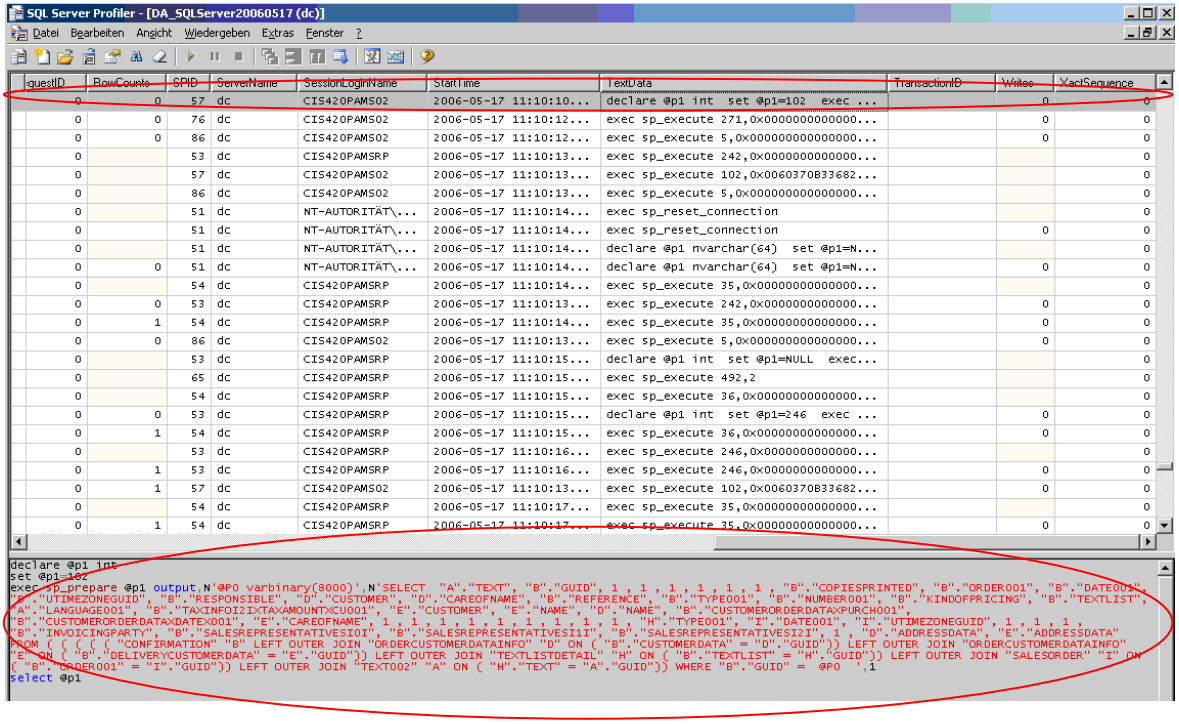


Abbildung 27 – *SQL-Server Profiler*, *declare* Statement.

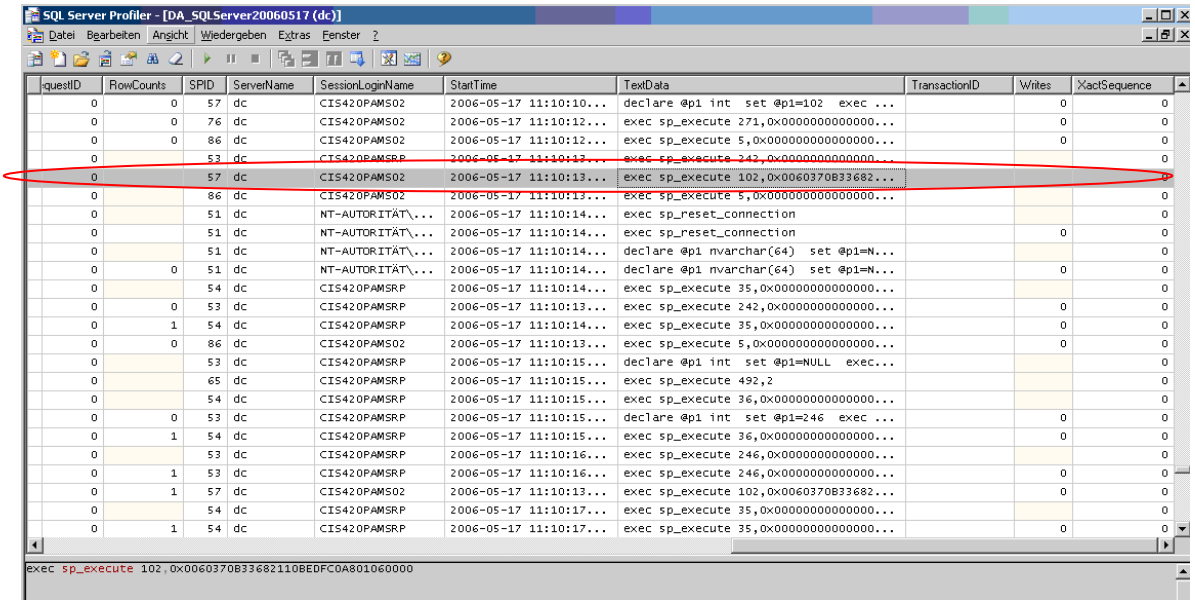


Abbildung 28 – *SQL-Server Profiler*, *execute* Statement.

DB2 UDB for iSeries

Die Suche nach teuren SQL-Statements kann für die DB2 UDB for iSeries über den *iSeries Navigator* durchgeführt werden (Bedoya, 2004). Über den Punkt *Datenbanken* kann unter der Auswahl *SQL Performance Monitors* ein neuer Monitor erstellt werden, der ähnlich dem *SQL-Server Profiler* alle Ereignisse an der Datenbank mitloggt. Durch Klick mit der rechten Maustaste auf den Monitor und Anwählen des Punktes *mit EXPLAIN bearbeitbare Anweisungen* bekommt

man eine Übersicht aller SQL-Anweisungen. Diese sind nun nach Verarbeitungszeit zu sortieren, siehe dazu Abbildung 29.

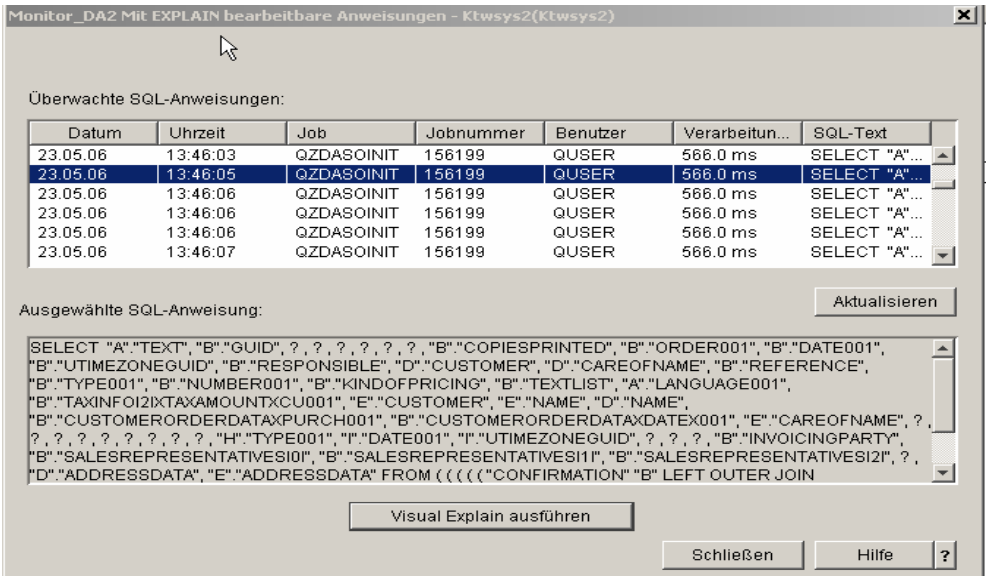


Abbildung 29 – *iSeries Navigator*, Auswahl *Mit EXPLAIN Plan bearbeitbare Anweisungen.*

Durch Betätigen des Buttons *Visual Explain ausführen* wird der Ausführungsplan angezeigt. Durch die Auswahl von *Aktionen/Advisor* in der Anzeige des Ausführungsplans wird der *Advisor* gestartet. Dieser schlägt Indizes vor, die in der Anwendung gleich erstellt werden können. Über den Punkt *Datenbanken* und die Auswahl *Schemata/Indizes* im *iSeries Navigator* kann die Verwendung der Indizes geprüft werden.

Auch auf der DB2 UDB for iSeries wurde der Vorgang mit dem Bericht *Auftragsbestätigung* durchgeführt, siehe Abbildung 29. Der Ausführungsplan führte ursprünglich einen Full Table Scan auf alle an der Abfrage beteiligten Tabellen aus. Der *Advisor* hat deshalb die Erstellung von Indizes auf die GUID von fünf beteiligten Tabellen vorgeschlagen. Nach Erstellung der fünf Indizes hat sich die Ausführungszeit des Statements von 566ms auf 299ms verbessert. Ein erneutes Aufrufen des Ausführungsplanes bestätigte die Verwendung der Indizes, es wurde kein Full Table Scan mehr durchgeführt (ohne Abbildung, da der Ausführungsplan mehrere Seiten beinhaltet). Die fünf Indizes sollten nun also auch in Semiramis erstellt werden.

Eine weitere hilfreiche Möglichkeit ist es, sich das letzte SQL-Statement eines Datenbankjobs anzeigen zu lassen. Dazu geht man über den Punkt *Ablaufsteuerung* in die Auswahl *aktive Jobs* und sucht nach einem Datenbankjob mit hoher CPU-Last. Durch Klick mit der rechten Maustaste und Auswahl von *Details/letzte SQL-Anweisung* bekommt man die letzte SQL-Anweisung dieses Jobs angezeigt. Mit dieser Methode können SQL-Anweisungen gefunden werden, die entweder sehr lange dauern oder hängen, beispielsweise durch Deadlocks verursacht.

3.3.4.3 Identifizierung von Schreib/Lese (I/O) Problemen

Stellt man fest, dass Lese- und Schreibvorgänge sehr langsam sind, oder I/O-Warteschlangen sehr lang sind (siehe Abschnitt 3.3.5), dann kann es Vorteile bringen, häufig verwendete Dateien auf unterschiedliche Datenträger zu legen. Dazu muss eine Analyse der Verteilung der I/O-Last

durchgeführt werden. Stellt man fest, dass beispielsweise zwei Datenbankfiles häufig gelesen oder geschrieben werden, die sich auf einem Datenträger befinden, so kann die Verteilung dieser beiden Dateien auf unterschiedliche Platten eine Performanceverbesserung bringen. (Schneider, 2005, S. 98). Im *Oracle Enterprise Manager* kann die Verteilung der I/O-Last unter dem Punkt *File IO Stats* eingesehen werden. Beim SQL-Server konnte keine solche Anzeige ermittelt werden. Für die DB2 UDB for iSeries ist die Analyse der I/O-Last nicht relevant, da unter dem dazugehörigen Betriebssystem i5/OS die Verwaltung der Datenbankdateien direkt dem Betriebssystem unterliegt (Siehe Abschnitt 2.2.1.4.).

3.3.4.4 Identifizierung von Sperrproblemen

Sperren beeinflussen die Performance eines DBMS negativ. Deshalb sollten sie so selten wie möglich auftreten. Treten Sperren über längere Zeiträume oder Deadlocks auf (d.h. sie konnten vom DBMS nicht beseitigt werden), können folgende Maßnahmen zur Beseitigung der Sperre oder des Deadlocks führen: Wenn die Sperre von einem Programm gehalten wird, das offensichtlich nicht von Semiramis stammt (z.B. eine externe Abfrage mit Microsoft Query, die Kreuzprodukte beinhaltet, und deshalb sehr lange dauert, oder ein Prozess, der nicht ordnungsgemäß beendet wurde), kann es beendet werden. Wird eine Sperre von einem Programm über mehrere Minuten gehalten, so ist Rücksprache mit dem Benutzer zu nehmen, damit dieser die Anwendung beendet, die er gerade bearbeitet. Trifft keiner der oben genannten Punkte zu, so kann es durch generelle Performanceprobleme der Datenbank vorkommen, dass Sperren lange gehalten werden. In diesem Fall ist zuerst das Performanceproblem zu beheben (Schneider, 2005). Eine hohe Anzahl an gehaltenen Sperren kann aber auch durch schlecht programmierte Anwendungen hervorgerufen werden.

Für die DB2 UDB for iSeries kann in diesem Zusammenhang der Befehl *wrkobjlck* verwendet werden (Bartley, 2005). Dieser zeigt die gehaltenen Sperren und die Art der Sperre pro Job an.

Die wichtigsten Kennzahlen für die Analyse von Sperren des SQL-Servers finden sich im Objekt *SQLServer:Sperren* des *perfmon* (Schwartz, 2005):

- „Sperranforderungen/Sekunden“, „Sperrtimeouts/Sekunde“, „Anzahl der Deadlocks/Sekunde“: Alle diese Werte sollten so niedrig wie möglich sein, die Sperrtimeouts und die Anzahl der Deadlocks (hier sind „echte“ Deadlocks, also gegenseitiges Blockieren zweier Prozesse, wie in Abschnitt 2.2.2.4 beschrieben, gemeint) am Besten 0.

Für Oracle kann in diesem Zusammenhang die View *V$LOCK* abgefragt werden, mit der Abfrage *SELECT * from V$LOCK WHERE request > 0* (Chan, 2005).

3.3.4.5 Sonstige Tuning Maßnahmen bzw. Monitoring Möglichkeiten

Weitere aussagekräftige Kennzahlen von Datenbanksystem sind so genannte Wait-Statistiken. Wartevorgänge können beispielsweise durch Ereignisse wie nicht verfügbare Speicherzuweisungen oder durch das vorherige Setzen einer Sperre verursacht werden. Die Gesamtwartezeit ist nun

relativ zum Betrachtungszeitraum zu sehen. Wenn die Gesamtsumme bei einem Zeitraum von 45 Minuten 30min beträgt, ist der Wert sehr hoch. Wenn bei gleichem Betrachtungszeitraum die Gesamtsumme nur 30s beträgt, ist der Wert sehr niedrig (Chan, 2005).

Der Befehl *wrkactjob* für die iSeries bietet in der Zeile „Status“ Informationen über den Status eines Prozesses. Ein solcher Status kann z.B. „Lockwait“ (LCKW) sein. Durch Auswahl der Option *11* kann angezeigt werden, welche Sperren auf diesen Job gehalten werden (Bartley, 2005). Eine weitere interessante Kennzahl, die die iSeries bietet, ist die „CPU-Auslastung (Datenbankkapazität)“ in der *Systemüberwachung*. Siehe dazu auch Abschnitt 3.3.5. Dieser Wert zeigt die CPU-Auslastung der Datenbankjobs an, und ist deshalb interessant, da bei einer Semiramis Installation auf der iSeries Applikations- und Datenbankserver auf der gleichen Maschine laufen.

Für den SQL-Server gibt es im *perfmon* das Objekt *SQLServer:Wartestatistik*, mit unterschiedlichen Kennzahlen je nach Art des Wartevorgangs. Weitere interessante Kennzahlen, die der *perfmon* bietet, sind unter Anderem (Schwartz, 2005):

- „Batchanforderungen/Sekunde“.
- „SQL-Kompilierungen/Sekunde“: Anzahl der Kompilierungen pro Sekunde. Dieser Wert sollte so klein wie möglich sein, maximal 40% der Batchanforderungen.
- „Erneute SQL-Kompilierungen/Sekunde“: Neukompilierung von *stored procedures*. Dieser Wert sollte so klein wie möglich sein, da *stored procedures* so gestaltet sein sollen, dass sie nur einmal kompiliert werden müssen.

Für Oracle gibt es im *Enterprise Manager* eine große Zahl an Kennzahlen betreffend Wait-Statistiken. Die Wait-Statistiken werden von einem Thread erhöht, wenn er auf ein Ereignis warten musste. Diese Summe an Wartevorgängen wird pro Ereignistyp mit der Wartezeit multipliziert, und man erhält eine Gesamtwartezeit pro Ereignis (Chan, 2005). Weitere interessante Kenzahlen bietet der *Oracle Enterprise Manager* auf der *Performance Page*. Diese gibt einen grafischen Überblick über den Zustand des Systems. Dabei werden unter Anderem die CPU-Auslastung des Hosts, durchschnittliche aktive Sessions, Platten I/O’s und der Instance-Durchsatz angezeigt. Weiters bietet der *Oracle Enterprise Manager* Funktionen um Berichte auszudrucken. Diese können sowohl für den aktuellen Zeitpunkt, wie auch für gewählte Zeitpunkte in der Vergangenheit (Snapshots) ausgegeben werden. Diese Berichte sind sehr umfangreich und zeigen unter Anderem Top SQL-Statements gewichtet nach unterschiedlichen Kriterien (Ausführungszeit, Lesezugriffe, CPU Zeit, usw.).

3.3.4.6 Zusammenfassung

In Abschnitt 3.3.4.2 sind starke Unterschiede zwischen den einzelnen Tuning Advisors der drei Datenbank-Plattformen aufgefallen. Jeder Tuning Advisor hat hier die Erstellung von sehr unterschiedlichen Indizes für dasselbe Datenbank-Statement vorgeschlagen. Unter Oracle wurde die Erstellung von zwei Indizes empfohlen, der SQL-Server hat kein Optimierungspotential gefunden, während auf der DB2 UDB for iSeries fünf Indizes vorgeschlagen wurden. Der Grund könnte folgender sein: Alle drei Datenbankplattformen verwenden einen Cost Based Optimizer.

Dieser erstellt Statistiken für Tabellen und Indizes und berechnet unter Berücksichtigung der aktuellen Systemauslastung die Kosten für die einzelnen Aktionen des aktuell erstellten Zugriffsplans. Da die Systeme unterschiedliche Auslastung hatten (aufgrund unterschiedlicher Hardware-Ausstattung), könnte der CBO dadurch beeinflusst worden sein. Aber auch die unterschiedliche Architektur der drei DMBS hat hier natürlich Einfluss.

Abbildung 30 stellt eine Zusammenfassung der Vorgehensweisen für die Datenbankanalyse dar. Sie wird im Weiteren zur Erstellung der Dokumentenvorlage verwendet.

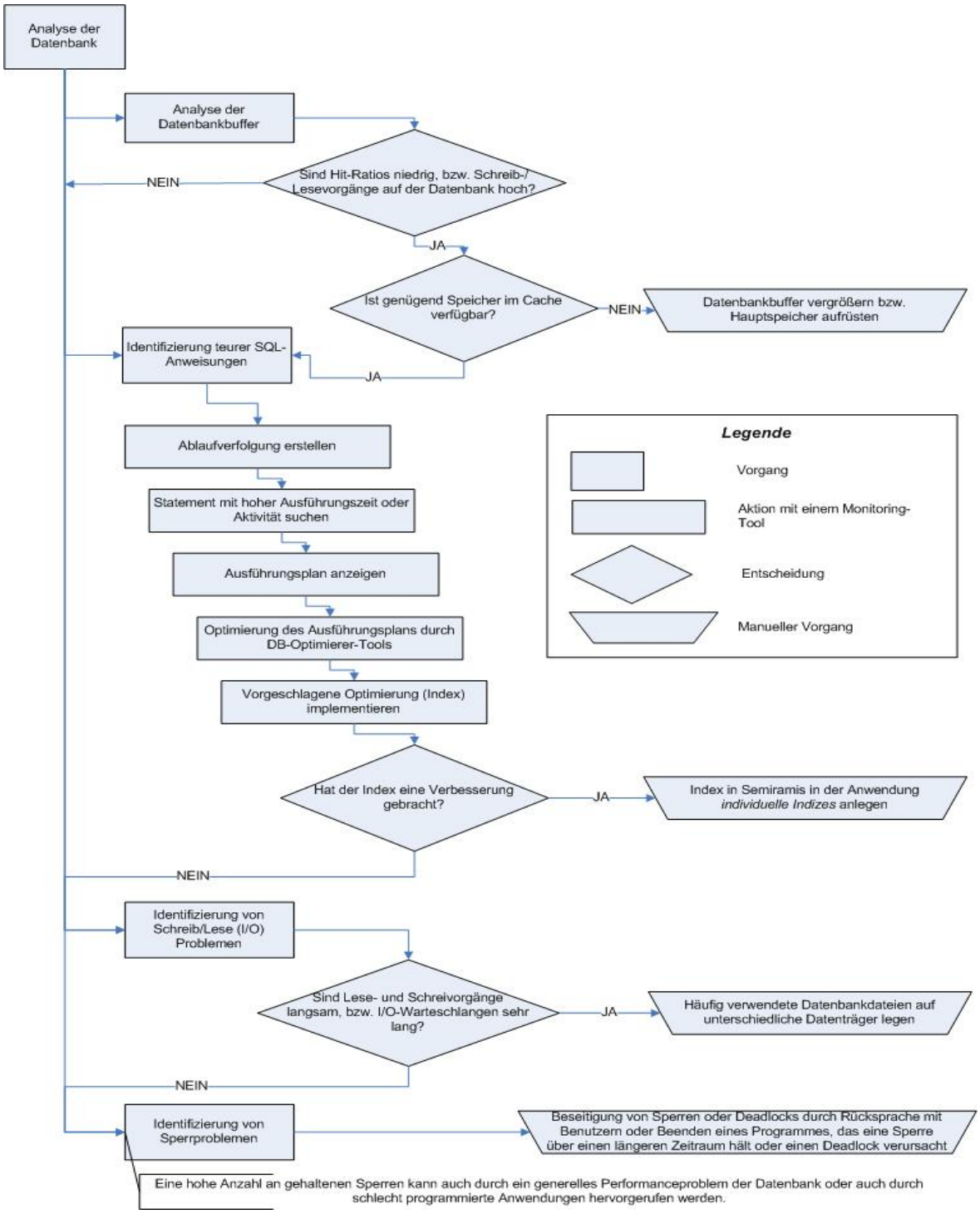


Abbildung 30 – Zusammenfassung der Datenbank-Analyse.

3.3.5 Hardware

Haben Tuning Maßnahmen keine sichtbaren Erfolge gezeigt, bzw. ist man sich von Vornherein sicher, dass der Fehler nicht an Semiramis oder der Datenbank liegt, folgt die Analyse der Hardware betreffend CPU, Speicher und I/O. Zu beachten ist hier, dass ein Semiramis-System üblicherweise auf mehrere Server verteilt ist. So sind normalerweise ein oder mehrere Server für die Datenbank, SAS und für den SOM vorhanden. Auf einer iSeries kann Datenbank und SAS auf der gleichen Maschine installiert sein. Somit müssen auch mehrere Server auf ein Hardwareproblem untersucht werden. Um einen Engpass der CPU festzustellen, sind Kennzahlen wie CPU-Auslastung, Interrupts und die durchschnittliche Warteschlangenlänge zu analysieren. Um ein Speicherproblem festzustellen, sind der verfügbare physische bzw. virtuelle Speicher und die Seitenfehlerrate aufschlussreich. Für die Identifizierung eines I/O-Problems sind Kennzahlen wie freier Speicherplatz, Auslastung, mittlere Warteschlangenlänge, und Schreib-/Lesevorgänge auf Festplatten bzw. RAID-Systeme zu analysieren. Jedes Betriebssystem bietet hier andere Kennzahlen bzw. Tools an. Nachfolgend eine Auflistung aller relevanten Kennzahlen und Tools für die einzelnen Betriebssysteme.

3.3.5.1 Windows

Um einen schnellen Überblick über die Performance eines Windows Systems zu erhalten, eignet sich der *Task Manager*. Will man detailliertere oder historische Daten analysieren, so eignet sich der *perfmon*. Die meisten Kennzahlen des *Task Managers* können auch vom *perfmon* angezeigt werden, deshalb wird hier der *perfmon* näher betrachtet. Unterschiede zwischen *Task Manager* und *perfmon*, sowie eine Anleitung zum *perfmon* und eine Beschreibung der Kennzahlen, die von Microsoft empfohlen werden, dem „Minimum Set of System Counters“, finden sich in (Microsoft, 2006).

Der *perfmon* besitzt Objekte, die wiederum aus Performance-Countern bestehen. Einem Performance Monitor können beliebig viele Counter hinzugefügt werden. Echtzeit-Daten werden von rechts nach links aktualisiert, ein Strich in der Mitte zeigt den aktuellen Zeitpunkt an. Nachfolgend eine Aufstellung der wichtigsten Kenzahlen für CPU, Speicher und I/O.

CPU

Kennzahlen über die CPU finden sich in den Objekten *Prozessor* und *Serverwarteschlangen*.

- „Prozessorzeit (%)“: Die aktive Zeit (busy time) des Prozessors wird angezeigt. Der *Task Manager* zeigt den prozentualen Anteil eines Prozesses an der CPU-Zeit an. Zusätzlich gibt es die Kennzahlen „Benutzerzeit (%)“ und „Privilegierte Zeit (%)“, welche den Anteil der Benutzer- bzw. der Kernelprozesse an der CPU-Zeit anzeigen. Die Kennzahl „Leerlaufzeit (%)“ gibt die inaktive Zeit des Prozessors in Prozent an.
- „Interrupts/s“: Zeigt die Anzahl von Hardware Interrupts pro Sekunde an.
- „Warteschlangenlänge“: Die durchschnittliche Warteschlangenlänge der CPU wird hier angezeigt.

Speicher

Kennzahlen über den Speicher finden sich in den Objekten *Speicher* und *Auslagerungsdatei*.

- „Belegung (%)“: Zeigt die prozentuelle Verwendung der Auslagerungsdatei an.
- „Seiten/s“: Zeigt die Anzahl der Seiten an, die pro Sekunde ein- bzw. ausgelagert werden.
- „Verfügbare MB“: Zeigt den verfügbaren physikalischer Speicher an.

I/O

- I/O Kennzahlen finden sich in den Objekten *Logischer Datenträger* und *Physikalischer Datenträger*. Das Objekt *Logischer Datenträger* kann auch RAID-Systeme analysieren. Bei den meisten der Kennzahlen in diesen beiden Objekten können alle oder einzelne Festplatten gewählt werden.
- „Freier Speicherplatz (%)“: Zeigt den freien Speicherplatz der Datenträger an.
- „Zeit (%)“: Die aktive Zeit (busy time) der Datenträger wird angezeigt.
- „Durchschnittliche Warteschlangenlänge des Datenträgers“.
- „Lesevorgänge/s“.
- „Schreibvorgänge/s“.

3.3.5.2 Linux

Linux bietet eine Vielzahl von Kommandozeilenbefehlen zur Performanceanalyse an, siehe dazu (Anderson, 2005).

CPU

top: Der Befehl *top* ist eine Echtzeit-Anzeige von Prozessen, sortiert nach CPU-Auslastung. Die Auslastung wird in die Prozesszustände system (Kernelprozesse), user (Benutzerprozesse), idle (CPU ist untätig) und nice (Reduzierung der Priorität, siehe Abschnitt 2.2.1.4) aufgeteilt.

Wichtige Anzeigen von *top* sind:

- *Load Average*: Diese drei Felder zeigen die Anzahl der lauffähigen Prozesse in der CPU-Warteschlange über die letzte Minute, die letzten fünf Minuten und die letzten 15 Minuten an.
- *CPU-States*: Zeigt die Auslastung der vier möglichen CPU Zustände an.
- *Mem & Swap*: Diese zwei Zeilen zeigen an, wie viel realer und wie viel Swap Speicher verfügbar, benutzt und frei ist. Auch wird angezeigt, wie viel Speicher für Buffer und Caches verfügbar ist.
- *Process Information*: Für alle Prozesse werden die Prozessnummer (PID), der Benutzer, die Priorität, der *nice* Wert, die Größe, RSS (Gesamtmenge an physischem Speicher),

Shared Memory, Status, %CPU, %Speicher, CPU Zeit seit der Prozess gestartet wurde, und das Kommando, das den Prozess gestartet hat, angezeigt.

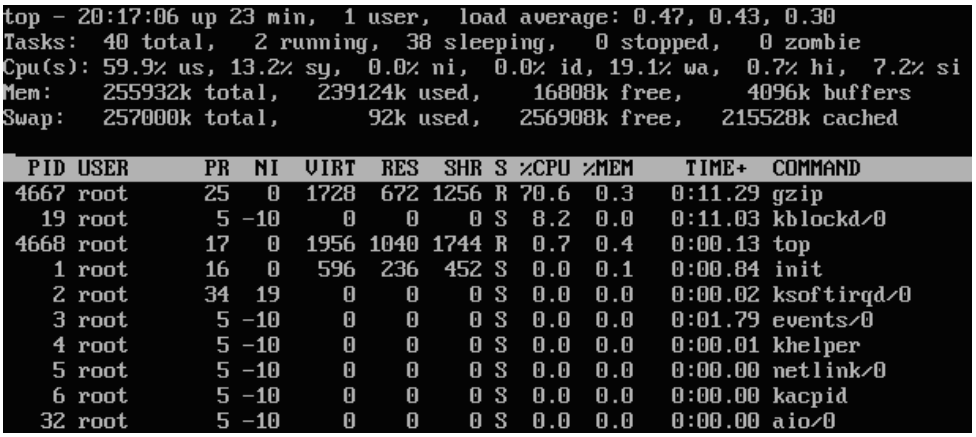


Abbildung 31 – Ausgabe des Befehls *top*.

Speicher

vmstat: Der Befehl *vmstat* liefert Informationen über den virtuellen Speicher und schließt sich selbst von den angezeigten Statistiken aus. Wichtige Anzeigen von *vmstat* sind:

- *procs*: *r* zeigt die Anzahl der wartenden, *w* die Anzahl ausgelagerten (in Abbildung 32 nicht angezeigt) und *b* die Anzahl der Prozesse in uninterruptible sleep (Prozesse, die z.B: auf eine I/O-Operation warten, die aber unter keinen Umständen abgebrochen werden dürfen) an.
- *Memory*: Gleiche Anzeige wie bei *top*: Swap, Free, Buffer und Cache Speicher.
- *Swap*: Zeigt Seitenein- bzw. Auslagerungen an.
- *I/O:* Zeigt Blöcke, die gelesen und geschrieben werden.
- *System*: Zeigt Interrupts und Context Switches pro Sekunde.
- *CPU*: Gleiche Anzeige wie bei *top*: Prozesse im user, system und idle Status.

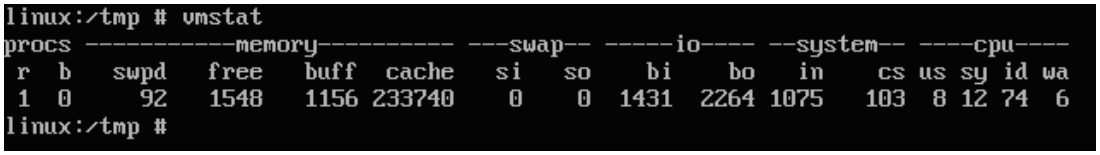


Abbildung 32 – Ausgabe des Befehls *vmstat*.

Disk

iostat: Der Befehl *iostat* generiert zwei Reports: CPU Aktivität, und die jeweilige I/O-Geräteauslastung.

```
linux:/tmp # iostat
Linux 2.6.8-24-default (linux)  05/16/06

avg-cpu:  %user   %nice    %sys %iowait   %idle
           8.77    0.07   12.07    8.42   70.67

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
fd0               0.00         0.00         0.00          4          0
sda              87.86      3209.60      5784.93    5127406    9241544
hdc               0.01         0.02         0.00         32          0
```

Abbildung 33 – Ausgabe des Befehls *iostat*.

df: Der Befehl *df –k –al zeigt* den freien Speicherplatz in kB aller Datenträger an**.**

Sonstiges

procinfo: Mit dem Befehl *procinfo –fn30* können Informationen über Speicher, CPU und I/O abgefragt werden. Auch die Summe aller Interrupts pro Interrupt Request (IRQ) können angezeigt werden. Mit Drücken der *Enter* Taste werden Daten aktualisiert.

```
Linux 2.6.8-24-default (geeko@buildhost) (gcc 3.3.4 ) #1 Wed Oct 6 09:16:23 UTC 2004 1CPU [linux.]
[unknown command]
Memory:      Total        Used        Free      Shared     Buffers
Mem:        126100       66588       59512           0       15312
Swap:       257000           0      257000

Bootup: Tue May 30 04:34:12 2006    Load average: 0.00 0.00 0.00 1/38 4519

user  :       0:00:00.83   0.3%  page in :        0
nice  :       0:00:00.00   0.0%  page out:        0
system:       0:00:03.88   1.6%  swap in :        0
idle  :       0:03:57.10  95.5%  swap out:        0
uptime:       0:41:30.94         context :    94110

irq  0:   2483089 timer               irq  9:        0 acpi, uhci_hcd, Enso
irq  1:       619 i8042               irq 11:     4302 BusLogic BT-958
irq  2:         0 cascade [4]         irq 12:     2167 i8042
irq  6:         5                     irq 15:       28 ide1
```

Abbildung 34 – Ausgabe des Befehls *procinfo*.

sar: Der Befehl *sar* ist das kommandozeilenbasierte Gegenstück zu *perfmon* unter Windows, und bietet die Möglichkeit, alle Kennzahlen der oben genannten Befehle in einem Logfile aufzuzeichnen und auszuwerten. Auswertungen können in eine csv-Datei umgewandelt, und mit Microsoft Excel ausgewertet werden oder in ein relationales Datenbank-System importiert werden. Bevor man mit *sar* arbeiten kann, muss ein Logfile aufgezeichnet werden. Dieses Aufzeichnen sollte am Besten täglich mittels eines *cron*-Jobs geschehen, mit jeweils einer Datei pro Tag, die zyklisch überschrieben wird. Der Befehl für das Aufzeichnen des Logfiles lautet: *sar -o /tmp/last2sar.log interval count* (*interval* ist der Aktualisierungsintervall in Sekunden, *count* die Anzahl der Aufzeichnungen. Will man einen Tag mit einem Aktualisierungsintervall von einer Minute aufzeichnen, muss *interval* 60 und *count* 1440 sein). Nachfolgend einige Beispiele für die Anwendung von *sar*, weiterführende Hinweise finden sich in (Linux, 2006; Anderson, 2005).

- Speicherkennzahlen von 20:41 – 20:42:
    - Speicherauslastung: *sar -r -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00*

```
linux:/tmp # sar -r -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00
Linux 2.6.8-24-default (linux)  05/16/06

20:41:01    kbmemfree kbmemused  %memused kbbuffers  kbcached kbswpfree kbswpused  %swpused  kbswpca
d
20:41:06         2868    253064     98.88      4840    228432    256908        92      0.04
0
20:41:11         1584    254348     99.38      4944    229684    256908        92      0.04
0
20:41:16         2984    252948     98.83      5068    227992    256908        92      0.04
0
20:41:21         2900    253032     98.87      5020    228164    256908        92      0.04
0
20:41:26        50764    205168     80.17      5112    180164    256908        92      0.04
0
20:41:31        99844    156088     60.99      5268    131000    256908        92      0.04
0
20:41:36        15964    239968     93.76      5588    214724    256908        92      0.04
0
20:41:41         1504    254428     99.41      2392    231956    256908        92      0.04
0
20:41:46         1676    254256     99.35      2624    231716    256908        92      0.04
0
20:41:51         1660    254272     99.35      1316    233008    256908        92      0.04
0
20:41:56        91324    164608     64.32      1472    143420    256908        92      0.04
0
Average:        24825    231107     90.30      3968    207296    256908        92      0.04
0
```

Abbildung 35 – Befehl *sar* mit Ausgabe für Speicherauslastung.

- Paging: *sar -B -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00*

```
linux:/tmp # sar -B -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00
Linux 2.6.8-24-default (linux)  05/16/06

20:41:01     pgpgin/s pgpgout/s   fault/s  majflt/s
20:41:06      8027.36   1213.68      2.21      0.00
20:41:11      9178.40   1286.40      2.20      0.00
20:41:16     11715.43   1462.12      2.20      0.00
20:41:21     12345.71   1822.75      2.20      0.00
20:41:26      3177.51   2093.98      2.21      0.00
20:41:31      2634.54   2051.41      2.21      0.00
20:41:36      1971.89  12844.98     63.86      0.80
20:41:41      8328.80  10248.80     39.40      0.20
20:41:46      7483.20   7208.80      2.20      0.00
20:41:51      7020.80   6021.60      2.20      0.00
20:41:56      7287.37   7169.54     34.87      0.20
Average:      7202.04   4857.49     14.15      0.11
```

Abbildung 36 – Befehl *sar* mit Ausgabe für Paging.

- CPU-Kennzahlen von 20:41 – 20:42 und Ausgabe in eine csv-Datei:
  - CPU-Auslastung: *sar -uHt -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00 > /tmp/testsarauswertung.csv*

```
linux:/tmp # sar -u -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00
Linux 2.6.8-24-default (linux)  05/16/06

20:41:01          CPU     %user     %nice   %system   %iowait     %idle
20:41:06          all     43.26      0.00     14.49     42.25      0.00
20:41:11          all     65.80      0.00     21.40     12.80      0.00
20:41:16          all     60.12      0.00     22.24     17.64      0.00
20:41:21          all     63.87      0.00     23.35     12.77      0.00
20:41:26          all     67.27      0.00     18.67     14.06      0.00
20:41:31          all     67.27      0.00     18.07     14.66      0.00
20:41:36          all      8.23      0.00     69.88     21.89      0.00
20:41:41          all     26.40      0.00     41.00     32.60      0.00
20:41:46          all     64.00      0.00     22.60     13.40      0.00
20:41:51          all     64.00      0.00     24.00     12.00      0.00
20:41:56          all     58.72      0.00     24.65     16.63      0.00
Average:          all     53.55      0.00     27.30     19.14      0.00
```

Abbildung 37 – Befehl *sar* mit Ausgabe für die CPU-Auslastung. Die Umlenkung der Ausgabe in eine Datei wurde hier weg gelassen.

- CPU-Warteschlangenlängen: *sar -qHt -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00 > /tmp/testsarauswertung.csv*

```
linux:/tmp # sar -q -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00
Linux 2.6.8-24-default (linux)  05/16/06

20:41:01      runq-sz  plist-sz   ldavg-1   ldavg-5  ldavg-15
20:41:06            1        42      0.97      0.36      0.39
20:41:11            1        42      0.97      0.37      0.40
20:41:16            0        42      0.97      0.38      0.40
20:41:21            1        42      0.97      0.39      0.40
20:41:26            1        42      0.98      0.40      0.41
20:41:31            1        42      0.98      0.41      0.41
20:41:36            1        42      0.98      0.42      0.41
20:41:41            1        42      0.98      0.43      0.42
20:41:46            1        42      0.98      0.44      0.42
20:41:51            1        42      0.98      0.45      0.42
20:41:56            1        42      1.06      0.47      0.43
Average:            1        42      0.98      0.41      0.41
```

Abbildung 38 – Befehl *sar* für Ausgabe von CPU-Warteschlangenlängen. Die Umlenkung der Ausgabe in eine Datei wurde hier weg gelassen.

- I/O Kennzahlen von 20:41 – 20:42 für Disk dev8-0: *sar -d -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00 |grep dev8-0*

In Abbildung 39 gibt die erste Spalte die Zeit an, die zweite die Disk, die dritte die Transferrate (Transfers pro Sekunde), die vierte die gelesenen, die fünfte die geschriebenen Sektoren, wobei in diesem Beispiel ein Sektor 512 Bytes entspricht.

```
linux:/tmp # sar -d -f /tmp/last2sar.log -s 20:41:00 -e 20:42:00 |grep dev8-0
20:41:06       dev8-0    158.75  16054.73   2427.36
20:41:11       dev8-0    183.60  18356.80   2572.80
20:41:16       dev8-0    229.26  23430.86   2924.25
20:41:21       dev8-0    243.51  24691.42   3645.51
20:41:26       dev8-0    102.61   6355.02   4187.95
20:41:31       dev8-0    100.20   5269.08   4102.81
20:41:36       dev8-0    265.66   3943.78  25689.96
20:41:41       dev8-0    382.80  16657.60  23715.20
20:41:46       dev8-0    294.20  14966.40  11200.00
20:41:51       dev8-0    255.20  14041.60  12043.20
20:41:56       dev8-0    273.15  14574.75  14339.08
Average:       dev8-0    226.38  14404.08   9714.97
```

Abbildung 39 – Befehl *sar* für Ausgabe von I/O-Kennzahlen.

3.3.5.3 i5/OS

Kennzahlen der iSeries können über eine *Systemüberwachung* im *Management Central* des *iSeries Navigator*, über den *Systemstatus* des *iSeries Navigator*, oder über die Befehle *wrkactjob*, *wrksyssts* und *wrkdsksts* abgefragt werden (Bartley, 2005; IBM, 2005; KTW, 2006).

Um eine *Systemüberwachung* zu erstellen, muss der Punkt *Überwachungen* und danach die Auswahl *System* im *iSeries Navigator Management Central* ausgewählt werden. Es kann nun eine neue *Systemüberwachung* erstellt werden. Es stehen ca. 25 Messgrößen zur Auswahl. Ist die Überwachung gestartet, werden Grafen zu den einzelnen Messgrößen angezeigt. Mit Klick auf

einen Grafen, z.B. CPU-Auslastung, werden die zehn Jobs angezeigt, die den höchsten Anteil an der CPU-Auslastung haben. Wie auch der *perfmon* unter Windows kann die *Systemüberwachung* der iSeries Echtzeit- und historische Daten anzeigen. Die Anzeige in der *Systemüberwachung* ist hierarchisch gegliedert. Im ersten Level werden systemweite Kennzahlen wie die CPU-Auslastung oder Disk-Auslastung angezeigt. Im zweiten Level werden die zugehörigen Kennzahlen angezeigt, im Beispiel der CPU die Jobs, die am meisten CPU-Zeit beanspruchen. Im dritten Level kann nun für jeden Prozess eine Detailansicht mit Kennzahlen wie der Priorität, oder dem Pool, in dem der Prozess läuft, angezeigt werden.

Der *Systemstatus* kann im *iSeries Navigator* durch Klick mit der rechten Maustaste auf den Servernamen, und der Auswahl *Systemstatus* angezeigt werden.

Nachfolgend eine Aufstellung der wichtigsten Kenzahlen der iSeries für CPU, Speicher und I/O:

CPU

Kennzahlen über die CPU können über eine *Systemüberwachung*, den *Systemstatus* oder den Befehl *wrkactjob* angezeigt *werden.*

- CPU Auslastung: Wird über den *Systemstatus*, im Reiter *Prozessoren* im Feld „CPU Belastung Allgemein“ angezeigt. Die CPU Auslastung pro Job zeigt der Punkt *Ablaufsteuerung* unter der Auswahl *aktive Jobs* im *iSeries Navigator*. In der *Systemüberwachung* finden sich weitere Kennzahlen zur CPU-Auslastung, wie „CPU-Auslastung durchschnittlich“, „interaktive Jobs“, „interaktive Funktionen“ oder „Datenbankkapazität“. Der Befehl *wrkactjob* zeigt im Wesentlichen dieselben Funktionen wie die aktiven Jobs im *iSeries Navigator*.
- Interrupts/s: Die Spalte „Act-Wait“ des Befehles *wrkssyssts* zeigt die Interrupts pro Sekunde für das jeweilige Subsystem an.
- Warteschlangenlänge: Im *iSeries Navigator* muss der Eintrag *Ablaufsteuerung* erweitert werden. Hier gibt es den Eintrag *Jobwarteschlangen*, mit dem alle Warteschlangen bzw. alle aktiven Warteschlangen des Systems angezeigt werden. Jobwarteschlangen sind Subsystemen zugeordnet. Es können mehrere Jobwarteschlangen pro Subsystem existieren. Pro Jobwarteschlange kann eine maximale Anzahl von möglichen Jobs eingestellt werden. Wenn ein Job in einer Warteschlange seines Subsystems nicht mehr Platz hat, dann wird er in eine neue Warteschlange mit sehr niedriger Priorität gereiht. Jedes Subsystem bekommt wiederum einen prozentuellen Anteil an der CPU. Jobwarteschlangen sind somit nicht identisch mit der Warteschlange der CPU. Sie sind deshalb auch nicht sehr aussagekräftig bezüglich des Gesamtsystems, mit Ausnahme der Identifizierung eines Jobs, der in eine neue Jobwarteschlange gestellt wird. Auch die in Abschnitt 2.2.1.4 gezeigten Prioritäten von Prozessen sind nicht identisch mit der Anzeige in den Jobwarteschlangen. Diese werden über den Befehl *wrkactjob* in der Spalte „Run Priority“ angezeigt.

Speicher

Kennzahlen über den Speicher können über den *Systemstatus* oder den Befehl *wrksyssts* angezeigt *werden.*

- Verwendung von temporärem Speicher: Über den Befehl *wrksyssts* können die Werte „Current unprotect used“ (momentan verwendeter temporärer Speicher) und „Maximum unprotect“ (maximal verwendeter temporärer Speicher seit Neustart) angezeigt werden. Im *Systemstatus*, Reiter *Plattenspeicherplatz* finden sich dieselben Informationen.
- Seiten/s: Im *Systemstatus* muss im Reiter *Arbeitsspeicher* der Button *aktive Speicherpools* gedrückt werden. Dann muss der gewünschte Pool mit Doppelklick ausgewählt werden. Im Reiter *Leistung* finden sich die Felder „Datenbankseiten/s“ bzw. „Nicht-Datenbankseiten/s“. Über den Befehl *wrksyssts* können die Seitenfehler pro Pool angezeigt werden (jeweils für Datenbank- und Nicht-Datenbankjobs).
- Verfügbarer physikalischer Speicher: Im *Systemstatus* muss im Reiter *Arbeitsspeicher* der Button *aktive Speicherpools* gedrückt werden. Dann muss der gewünschte Pool mit Doppelklick ausgewählt werden. Im Reiter *Konfiguration* zeigt das Feld „Aktuell“ die momentane Größe des Pools. Der Befehl *wrksyssts* zeigt in der Spalte „Pool Size M“ die aktuelle Größe des jeweiligen Pools. Es müssen nun die momentanen Größen aller Pools auf dem System zusammengezählt werden und dieser Wert mit der installierten Menge an physikalischem Speicher verglichen werden.

I/O

I/O Kennzahlen können über eine *Systemüberwachung*, den *Systemstatus* oder den Befehl *wrkdsksts* angezeigt werden.

- Freier Speicherplatz: Zeigt das Feld „Auslastung“ im Reiter *Plattenspeicherplatz* des *Systemstatus*.
- Aktive Zeit (busy time) der Datenträger: Im *Systemstatus* muss im Reiter *Plattenspeicherplatz* der Button *Plattenpools* gedrückt werden. Dann muss der gewünschte Pool mit Doppelklick ausgewählt werden. In der Spalte „%ausgelastet“ wird die Auslastung pro Festplatte angezeigt. Der Befehl *wrkdsksts* zeigt im Wesentlichen dieselben Informationen, hier zeigt die Spalte „%Used“ die Auslastung der jeweiligen Platte an. In der *Systemüberwachung* findet sich die Kennzahl „Plattenspeicher (durchschnittlich)“. Diese zeigt die Einzelauslastung einer Festplatte. Die Einzelauslastung einer Festplatte ist eine sehr wichtige Kennzahl, weil die iSeries Daten gleichmäßig auf alle Platten verteilt. Werden nun größere Datenmengen gelesen und ist eine Platte überlastet, dann dauert das Lesen sehr lange, da ja von allen Platten – auch von der überlasteten – gelesen wird. Deshalb gibt es auf der iSeries Tools, die Daten gleichmäßig auf Festplatten verteilen. Dies ist beispielsweise bei der Installation einer neuen Festplatte erforderlich.

- Plattenzugriffsarmauslastung: Der Befehl *wrkdsksts* zeigt in der Spalte „%Busy“ die Plattenzugriffsarmauslastung pro Festplatte an. In der *Systemüberwachung* findet sich die Kennzahl „Auslastung des Plattenzugriffsarms (durchschnittlich)“.
- Lesevorgänge: Der Befehl *wrkdsksts* zeigt in den Spalten „Read Rqs“ und „Read (K)“ die aktuelle Anzahl der Lesevorgänge bzw. die gelesene Datenmenge an.
- Schreibvorgänge: Der Befehl *wrkdsksts* zeigt in den Spalten „Write Rqs“ und „Write (K)“ die aktuelle Anzahl der Schreibvorgänge bzw. die geschriebene Datenmenge an.

3.3.5.4 Zusammenfassung

Es werden nun alle oben erwähnten Kennzahlen für die einzelnen Betriebssysteme gegenübergestellt. Die hier dargestellten Grenzwerte bzw. Handlungsempfehlungen konnten aufgrund von Befragungen erfahrener KTW-Mitarbeiter, einschlägiger Literatur (Microsoft, 2006; Schneider, 2005), den in dieser Arbeit vorgestellten theoretischen Grundlagen, sowie Praxiserfahrungen des Verfassers ermittelt werden. Deshalb ist diese Aufstellung als grobe Richtlinie zu sehen. Besonders die letzten beiden Spalten der folgenden drei Abbildungen, „Grenzwerte“ und „Kommentare/ Handlungsempfehlungen“ können sehr wertvolle Hinweise für eine Performanceanalyse darstellen.

CPU

Kennzahl	**Windows: *perfmon* , Task Manager**	**Linux: *top* , sar, vmstat**	**i5/OS: *iSeries Navigator,* wrkactjob, wrksyssts**	**Grenzwerte**	**Kommentare/ Handlungsempfehlungen**
CPU Auslastung	Prozessorzeit (%) Task Manager: CPU-Zeit pro Prozess	CPU states, %CPU pro Job sar: sar -u Ausgabe	Systemstatus: Feld CPU Belastung abgelaufen Systemüberwachung: CPU-Auslastung (durchschnittlich) wrkactjob bzw. aktive Jobs (CPU pro Job)	Windows: < 85% Linux: im Mittel zwischen 0 und 50% i5/OS: < 90%, CPU pro Job <40%	Finden und Analysieren der Prozesse, die einen hohen Anteil an der Prozessorzeit haben. Aufrüsten auf einen schnelleren Prozessor, oder Hinzufügen eines weiteren Prozessors.
Interrupts	Interrupts/s	vmstat: Sektion system, Spalte in	wrksyssts, Spalte Act-Wait	abhängig vom Prozessor. Linux, Windows: Für aktuelle CPU's < 1500 i5/OS: keine generelle Aussage zu treffen, da Hardware über IOP verwaltet wird	Eine dramatische Erhöhung dieser Kennzahl ohne korrespondierende Erhöhung der Systemaktivität lässt auf ein Hardwareproblem schließen. Identifizierung des Netzwerk-Adapters oder des Festplattencontrollers, der dieses Problem verursacht.
CPU-Warte-schlangen-länge	Warteschlangen-länge	load averages sar: sar -q Ausabe	Ablaufsteuerung /Jobwarteschlangen (Achtung: dies sind keine CPU Wartschlangen, sondern Jobwarteschlangen pro Subsystem)	Linux, Windows <4 i5/OS: Kontrolle, ob ein Job in einer weiteren Warteschlange mit niedriger Priorität vorhanden ist	Ein Erreichen dieses Grenzwertes kann einen Prozessor Flaschenhals signalisieren. Diese Kennzahl muss zur genaueren Aussage über einen längeren Zeitraum beobachtet werden.

Abbildung 40 – Gegenüberstellung wichtiger CPU-Kennzahlen. *Kursiv* geschrieben ist der Standard-Monitor, dieser wird in der Tabelle nicht explizit aufgeführt.

Speicher

Kennzahl	**Windows: *perfmon* , Task Manager**	**Linux: *vmstat,* sar, top**	**i5/OS: *iSeries Navigator,* wrksyssts**	**Grenzwerte**	**Kommentare/ Handlungsempfehlungen**
Belegung der Auslagerungs-datei bzw. des swap oder temporären Speichers	Belegung (%)	Memory Sektion, Spalte swpd, top: Swap Zeile sar: sar -B, Spalte kbswpused	Systemstatus, Reiter Plattenpeicher-platz, Sektion belegter temporärer Speicher wrksyssts: Current unprotect used, Maximum unprotect	Windows: < 70% Linux: <98% bei i5/OS nicht relevant, der temporäre Speicher kann einige hundert GB groß sein. Wenn der temporäre Speicher sehr stark über die Zeit anwächst, signalisiert dies ein Problem ("Speicherfresser-Anwendung").	Dieser Wert ist immer in Zusammenhang mit dem verfügbaren physikalischen Speicher zu sehen. Sinkt der verfügbare phsysikalische Speicher und kommt es zu einer Auslagerung, dann steigt der Wert. Die Auslagerungsdatei sollte in etwa 1,5 bis 2 mal so groß sein wie der physikalische Speicher.
Seiten/s	Seiten/s	sar: sar -B Ausgabe	Systemstatus: Reiter Arbeitsspeicher, Button aktive Speicherpools, gewünschten Pool mit Doppelklick auswählen, Reiter Leistung, Spalte Datenbanseiten/s bzw. Nicht-Datenbanseiten/s	Windows, Linux: < 20 i5/OS: < 10 (bei "warmer" JVM), für Datenbank-Pools kann dieser Wert kurzfristig auch höher sein, da die DB2 den verfügbaren Speicher voll ausnutzt, und so viele Seiten wie möglich im Speicher behält. Deshalb ist diese Kennzahl für Datenbanken nicht sehr aussagekräftig, da Seitenein- bzw. -auslagerungen unabhängig von gerade stattfindenden Datanbankabfragen stattfinden.	Ursachen für zu hohe Pagingraten sind vor allem zuwenig physikalischer Speicher, oder auch schlechte Seitenersetzungsstrategien von Applikationen (Beispielsweise eine falsche Caching-Strategie für ein Business Object in Semiramis).
Verfügbarer physikalischer Speicher	Verfügbare MB	memory Sektion, Spalte free top: Mem Zeile sar: sar -B, Spalte kbmemfree	Systemstatus: Reiter Arbeitsspeicher, Button aktive Speicherpools, gewünschten Pool mit Doppelklick auswählen, Reiter Konfiguration, Feld Aktuell wrksyssts: Pool Size M	Windows: > 4MB Linux: > 10% i5/OS: Es müssen die momentanen Größen aller Pools auf dem System zusammengezählt werden, und dieser Wert mit der installierten Menge an physikalischem Speicher verglichen werden.	Beenden von nicht benötigten Applikationen. Auslagern von Applikationen auf einen anderen Rechner. Prüfen, ob eine Applikation (z.B. eine Semiramis Anwendung) optimiert werden kann, damit sie weniger Speicher benötigt. Prüfen, ob nahezu der vollständige Java Heap bei einem "warmen" System verwendet wird. Wenn nein, Verkleinerung des Heaps. Wenn all dies nicht möglich ist, Hinzufügen von mehr physikalischem Speicher.

Abbildung 41 – Gegenüberstellung wichtiger Speicher-Kennzahlen. *Kursiv* geschrieben ist der Standard-Monitor, dieser wird in der Tabelle nicht explizit aufgeführt.

I/O

Kennzahl	**Windows:** ***perfmon***	**Linux:** ***iostat*** **, df, sar**	**i5/OS:** ***iSeries Navigator,*** **wrkdsksts**	**Grenzwerte**	**Kommentare/ Handlungsempfehlungen**
Freier Speicherplatz	Freier Speicherplatz (%)	df: df -k -al	Systemstatus: Reiter Plattenspeicherplatz, Feld Auslastung	> 15%	Installieren größerer oder weiterer Festplatten.
Auslastung	Zeit (%)	es konnten keine Linux native Tools zur Anzeige dieser Kennzahl ermittelt werden. Installation von Drittprodukten notwendig.	Systemstatus: Reiter Plattenspeicherplatz Button Plattenpools, gewünschten Pool mit Doppelklick auswählen, Spalte %ausgelastet wrkdsksts: % Used Systemüberwachung: Plattenspeicher (durchschnittlich).	Windows, Linux: < 90% i5/OS: <90%. Bei 99% schaltet sich das System ab Unterschiedliche Plattenauslastung ist sehr schlecht.	Ist die Last auf erhöhtes Paging zurückzuführen, dann Installation von mehr physikalischem Speicher. Ist die Anwendung naturgemäß I/O-lastig, dann Aufteilung der Last auf verschieden Festplatten oder Installation schnellerer Festplatten. i5/OS: Daten gleichmäßig auf alle Festplatten verteilen.
Plattenzugriffs-armauslastung	nicht verfügbar	nicht verfügbar	wrkdsksts: %Busy Systemüberwachung: Auslastung des Plattenzugriffsarms (durchschnittlich)	i5/OS: alle Zugriffsarme < 70% ein Zugriffsarm < 50%	Daten gleichmäßig auf alle Festplatten verteilen.
Durch-schnittliche Warte-schlangen-länge	Durchschnittliche Warteschlangen-länge des Datenträgers	es konnten keine Linux native Tools zur Anzeige dieser Kennzahl ermittelt werden. Installation von Drittprodukten notwendig.	nicht relevant, die Plattenzugriffsarm-auslastung ist aussgakräftiger.	Anzahl der Köpfe bzw. Spuren plus 2	Installieren schnellerer Festplatten mit mehr Köpfen.
Lese-vorgänge	Lese-vorgänge/s	Blk_read/s sar: sar- d, Spalte rd_sec/s (Achtung: Sektoren)	wrkdsksts: Read Rqs, Read (K)	Als Faustformel gelten 50-70 Lesevorgänge/s für moderne SCSI-Festplatten. Wenn die Festplatte über einen Cache Speicher verfügt, kann dieser Wert kurzfristig überschritten werden. Wird ein RAID verwendet, so ist der zusätzliche Overhead zu beachten.	Kontrolle der Spezifikationen der installierten Festplatten. Falls die durchschnittliche Datengröße von Lesevorgängen ermittelt werden kann, so kann die unter Abschnitt 2.2.1.3 beschriebene Berechnung zur Ermittlung der durchschnittlichen Zeit, um Daten von einer Festplatte abzurufen, angewendet werden. 1 dividiert durch den errechneten Wert ergibt die maximalen Lesevorgänge/s. Wenn der errechnete oder der Faustformel Wert überschritten wird, dann Installation schnellerer oder weiterer paralleler Festplatten.
Schreib-vorgänge	Schreib-vorgänge/s	Blk_wrtn/s sar: sar -d, Spalte wr_sec/s (Achtung: Sektoren)	wrkdsksts: Write Rqs, Write (K)	siehe Lesevorgänge/s	siehe Lesevorgänge/s

Abbildung 42 – Gegenüberstellung wichtiger I/O-Kennzahlen. *Kursiv* geschrieben ist der Standard-Monitor, dieser wird in der Tabelle nicht explizit aufgeführt.

3.3.6 Netzwerk

Das Monitoring auf Netzwerkseite gestaltet sich schwierig, da Semiramis ausschließlich über HTTPS kommuniziert, und die Daten dadurch verschlüsselt sind. Es können nur die Netzwerkleitungen generell auf ihre Auslastung überwacht werden. Dies reicht aber in der Regel aus. Wichtig ist es, die Netzwerkverbindungen groß genug zu dimensionieren, d.h. Gbit-Leitungen zwischen Servern, und 100Mbit-Leitungen zu den Clients. So treten Netzwerkprobleme in der Regel nicht auf. Wichtige Kommandozeilenbefehle zum Monitoring des Netzwerkes unter Windows sind:

- *ping*: Dient üblicherweise zur Überprüfung, ob ein Netzwerkgerät erreichbar ist oder nicht. Der Befehl *ping* kann aber auch für weitere Diagnosen herangezogen werden, beispielsweise für die Ermittlung der Path-MTU.
- *tracert*: zeichnet den Pfad eines Requests auf. Es werden alle durchlaufenden Subnets angegeben und auch die Zeit, die der Request benötigt, das jeweilige Subnet zu durchlaufen. So kann ein Subnet identifiziert werden das Probleme macht.

Beide Befehl, *ping* und *tracert* verwenden das Internet Control Message Protocol (ICMP), welches zum Austausch von Fehler- und Informationsmeldungen dient. Es kann vorkommen, dass Router aus Sicherheitsgründen dieses Protokoll sperren, deshalb kann es zu Verfälschungen kommen.

Der *perfmon* zeigt in den Objekten *Netzwerkschnittstelle*, *TCPv4*, *TCPv6*, *UDPv4*, *UDPv6, IPv4, IPv6, ICMP, ICMPv6, IPSec v4-Internetschlüsselaustausch* und *IPSec v4-Treiber* eine Vielzahl an Netzwerkkennzahlen an. Auf diese wird hier aber nicht näher eingegangen. Im Objekt *Netzwerkschnittstelle* sind die folgenden Kennzahlen von Bedeutung*:*

- „Aktuelle Bandbreite“: Ein geschätzter Wert für die aktuelle Bandbreite der Netzwerkschnittstelle in Bits pro Sekunde. Bei Schnittstellen mit fester Bandbreite, oder wenn keine zutreffende Schätzung gemacht werden kann, ist dieser Wert die Nennbandbreite.
- „Bytes gesendet/s“, „Empfangene Bytes/s“: Die Rate, zu der Bytes über jeden Netzwerkadapter gesendet bzw. empfangen werden, einschließlich Rahmencharakteristiken.

Alternativ können Produkte von Drittherstellern wie *Ethereal* – ein plattformunabhängiger Netzwerk-Sniffer – oder *NTOP* – dient zur Überwachung der Bandbreite – verwendet werden. Diese zwei genannten Tools – beides Open Source Produkte – müssen an zentraler Stelle, beispielsweise an einem Router installiert werden und können somit ein gesamtes Subnet überwachen.

3.3.6.1 Ermittlung der Path-MTU

Um die Path-MTU – die bei Semiramis speziell für VPN-Verbindungen von Außenstellen interessant ist – zu ermitteln, kann der Befehl *ping* verwendet werden. Es muss das *dont' fragment Flag* gesetzt werden, und die Länge des Paketes solange erhöht werden bis der Zielhost nicht mehr erreichbar ist. Der letzte Wert der Paketlänge bei dem der Ziel-Host noch erreichbar ist (das *dont'*

fragment Flag bewirkt, dass ein Paket das fragmentiert werden müsste, nicht mehr gesendet wird), entspricht der Path-MTU. Im Beispiel der Abbildung 43 ist die Path-MTU 1432. Die Implementierungen des *ping* Kommandos sind von Betriebssystem zu Betriebssystem unterschiedlich, als Beispiel wird hier nur die Implementierung unter Windows behandelt. Die Option *–f* setzt das *don't fragment* Flag, die Option *–l* gibt die Paketgröße an.

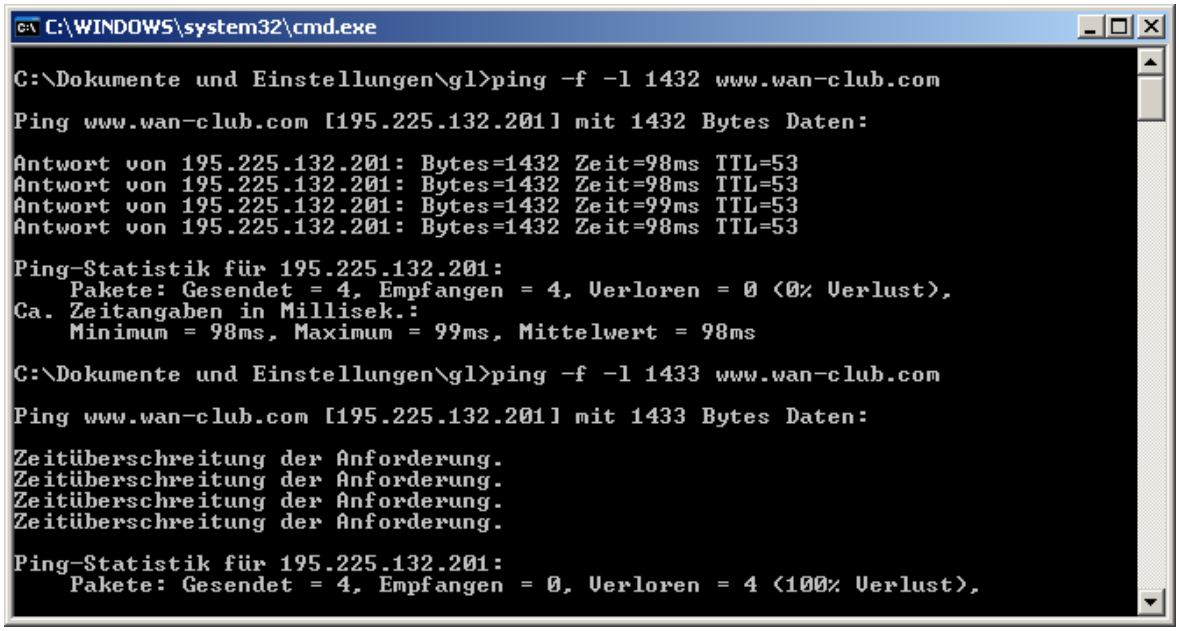


Abbildung 43 – Beispiel für die Ermittlung der Path-MTU unter Windows.

3.4 Präsentieren der Ergebnisse

Das Ergebnis der Analyse sollte ein Dokument sein, welches nach den Hauptabschnitten in Kapitel 3 (Zieldefinition, Dokumentation und Validierung der Systemparameter, Analyse der einzelnen Semiramis Komponenten und Präsentieren der Ergebnisse) gegliedert ist. Eine Dokumentvorlage dazu wird in Kapitel 4 beschrieben. Um das Lesen des Dokumentes zu Vereinfachen, sollten nach Möglichkeit unterstützende Grafiken verwendet werden. Beispielsweise können zeitliche Verläufe von Antwortzeiten oder CPU-Auslastungen in einer Grafik dargestellt werden. Nachfolgend sind die wichtigsten Punkte zusammengefasst, die das Verständnis von Grafiken verbessern können (Jain, 1991, S. 139ff).

- **Leseaufwand minimieren:** Der Leser der Grafik sollte minimalen Aufwand zum Verständnis der Grafik aufbringen müssen. Befinden sich mehrere Kurven auf einer Grafik, so ist direktes Beschriften der Kurven einer Legende vorzuziehen.
- **Information maximieren:** Beispielsweise sollen Axenbeschriftungen mit Wörtern anstatt mit Variablen erfolgen. Verwendet man Variablen, muss der Leser vorher im Dokument ihre Bedeutung nachlesen. Beschriftungen von Axen sollten so aussagekräftig wie möglich sein. Wenn man die tägliche CPU Auslastung darstellen will, soll die Axenbeschriftung auch genau so heißen.

- **Tinte sparen:** Unnötige Informationen sollen weg gelassen werden, beispielsweise sollen Axenlinien weg gelassen werden, sofern sie nicht unbedingt benötigt werden.
- **Geeignete Skalierung wählen:** Ein gutes Beispiel dazu ist die Darstellung der Verfügbarkeit eines Systems an sieben Tagen der Woche. Angenommen sie liegt immer knapp unter 100%. Wenn man nun die Verfügbarkeit in einem Balkendiagramm darstellt, bekommt man sieben Balken, die alle fast bis ans Ende der Skala reichen. Stellt man aber die Nicht-Verfügbarkeit dar, und skaliert die Grafik anders, bekommt man eine viel aussagekräftigere Grafik. Siehe dazu Abbildung 44.

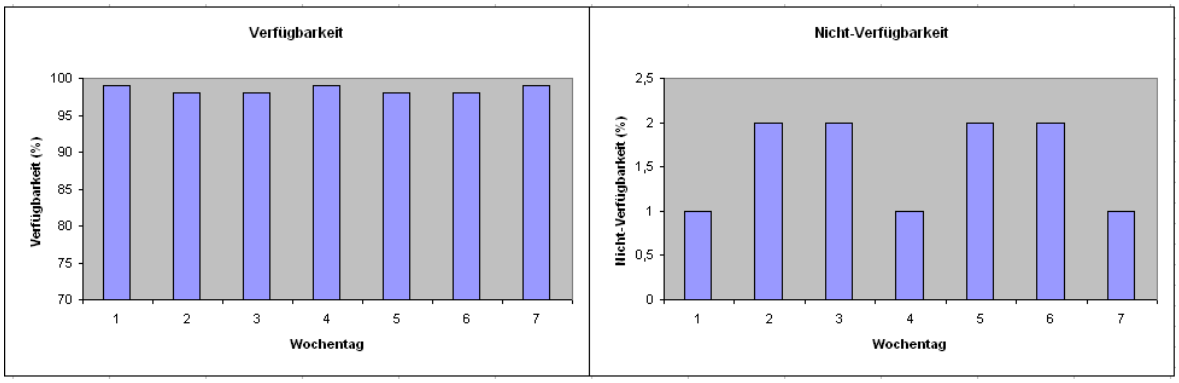


Abbildung 44 – Zwei Grafiken, die dieselben Daten darstellen (modifiziert nach: Jain, 1991, S. 142).

- **Gängige Praktiken verwenden:** Grafiken sollten das präsentieren, was ein Leser im Allgemeinen erwartet. So soll der Ursprung der Grafik (0,0) sein, die unabhängige Variable entlang der x-Achse, die abhängige Variable entlang der y-Achse dargestellt werden. Skalen sollten linear (oder in besondern Fällen logarithmisch) sein. Werte sollten von links nach rechts bzw. von unten nach oben ansteigen. Ausnahmen von dieser Regel sollten nur gemacht werden, wenn es keine andere Möglichkeit gibt, da sie immer einen Mehraufwand für den Leser bedeuten.
- **Mehrdeutigkeit vermeiden:** Axen, Skalen und Kurven sollen gekennzeichnet sein. In einem Chart sollten nach Möglichkeit nicht mehrere Variable dargestellt werden.

4 LÖSUNGSVORSCHLÄGE

Wie das vorherige Kapitel gezeigt hat, ist die Leistungsanalyse von Semiramis sehr komplex und es muss mit einer Vielzahl an unterschiedlichen Tools gearbeitet werden. In diesem Kapitel folgen vier Ansätze, wie die Analyse vereinheitlicht werden könnte. Auch finden sich hier Ansätze, wie man diese Lösungsvorschläge auf weitere ERP-Systeme anwenden könnte.

4.1 Tools von Drittherstellern verwenden

So genannte Enterprise Management Tools dienen zum Management der kompletten IT-Infrastruktur eines Unternehmens (Anderson, 2005a). Es können Betriebssysteme, Datenbanken

und Netzwerke überwacht werden. Einige dieser Tools bieten Unterstützung für gängige ERP-Systeme an. Sie könnten also prinzipiell zur Überwachung eines Semiramis-Systems herangezogen werden. Nachfolgend sind einige dieser Tools aufgelistet:

- *Silk Central Performance Manager* von segue Software (Segue, 2006).
- *AppManager Suite* von Net IQ (NetIQ, 2006).
- *OpenView Operations* von HP: Bietet Unterstützung für mySAP.com, Peoplesoft und Siebel (HP, 2006).
- *Tivoli Monitoring Applications* von IBM: Bietet Unterstützung für Siebel und SAP (Tivoli, 2006).
- *Microsoft Operations Manager* (MOM, 2006).

Der Nachteil der oben genannten Tools ist, dass sie sehr teuer sind. Außerdem stehen Semiramis spezifische Informationen wie z.B. zeitaufwändige Aktionen oder Daten der JVM nicht zur Verfügung. Auch die Unterstützung der iSeries ist, außer bei den Tivoli Monitoring Applications, nicht vorhanden. Datenbank-Tuning ist ebenso nicht möglich. Wenn also ein Kunde bereits eine große IT-Infrastruktur besitzt, macht es durchaus Sinn, ein solches Tool anzuschaffen, und die Semiramis-Systeme mit zu überwachen. Ein Enterprise Management Tool alleine für ein Semiramis-System anzuschaffen macht sich jedoch nicht bezahlt.

Eine Möglichkeit, die in einigen Jahren bestehen könnte – vorausgesetzt Semiramis verkauft sich in Zukunft so stark wie dies von der KTW Group gewünscht wird – ist, eine Kooperation beispielsweise mit IBM einzugehen, und die Tivoli Monitoring Applications auf Semiramis maß zuschneidern. IBM ist jetzt schon Partner der KTW, es könnte also durchaus Interesse bestehen.

4.2 Erweiterung *OpenNMS*

Eine weitere Lösungsvariante wäre die Erweiterung von *OpenNMS* auf die wichtigsten oben genannten Kennzahlen von Betriebssystemen und Datenbanken, sowie auf die Abfrage der Leistungsinformationen des Semiramis-Leistungsmonitorings. Alle behandelten Betriebssystem- und Datenbank-Plattformen bieten SNMP Unterstützung. Jedoch muss bei den meisten Plattformen eine zusätzliche Erweiterung mit installiert werden. Ebenso muss für den Semiramis SAS der überwacht wird, eine Erweiterung installiert werden. Auch die Installation des Programms selber ist aufwändig. Deshalb eignet sich *OpenNMS* für das Monitoring von KTW in-house Systemen, für Kundensysteme kommt es eher nicht in Frage. Was *OpenNMS* nicht ersetzen kann, sind Datenbank-Tuning Werkzeuge. Der Vorteil von *OpenNMS* ist, dass es – da Open Source – kostenfrei ist.

4.3 Integration in Semiramis

Die Integration der wichtigsten oben genannten Kennzahlen von Datenbanken, Betriebssystemen und der JVM sollte prinzipiell möglich sein, da Semiramis ja ohnehin darauf Zugriff hat. Da von der C.I.S. AG ein spürbares Interesse in dieser Richtung besteht (siehe die neuen Monitoring-

Möglichkeiten in Semiramis Version 4.2), würde auch eine Erweiterung möglicherweise Anklang finden. SAP bietet beispielsweise solche Integrationen an (Schneider, 2005). Für Semiramis würde sich eine Erweiterung des *Systemcockipts* um neue Karteireiter bzw. eine Erweiterung bestehender Karteireiter anbieten:

- Erweiterung des Karteireiters *Application-Server* um weitere Daten der JVM, wie detailliertere Informationen zum Heap (Eden, Tenured, Survivor, Permanent Space) und der Anzahl bzw. Ausführungsdauer der Garbage Collections.
- Erweiterung des Karteireiters *Datenbankverbindungen* um Informationen über Datenbankbuffer (Hit-Ratios, freier Speicher), Schreib-/Lesevorgänge auf Datenbankdateien und Datenbanksperren bzw. Deadlocks.
- Hinzufügen eines neuen Karteireiters *Betriebssystemmonitor*. Dieser müsste eine Filterung auf die unterschiedlichen Server in der Semiramis-Systemlandschaft ermöglichen. Diese Server müssten in die Kategorien Datenbank, SAS und SOM eingeteilt werden. In jeweils eigenen Bereichen könnten Kennzahlen zur CPU (Auslastung, Interrupts, durchschnittliche Warteschlangenlänge), Speicher (verfügbarer Speicher, Seitenfehler) und I/O (freier Speicherplatz, Auslastung, durchschnittliche Warteschlangenlänge, Schreib- bzw. Lesevorgänge) angezeigt werden.
- Hinzufügen eines neuen Karteireiters *Netzwerkmonitor*. Hier würden die Auslastung, die gesendeten bzw. empfangenen Bytes und die Anzahl der Kollisionen bzw. Fehler der Netzwerkinterfaces der einzelnen Server in der Semiramis-Systemlandschaft angezeigt werden. Hier müsste wiederum eine Filterung auf die unterschiedlichen Server in der Semiramis-Systemlandschaft möglich sein. Eine weitere sehr hilfreiche Funktion wäre eine automatische Path-MTU Ermittlung. So könnte man sich zum Beispiel von einer Außenstelle an ein zentrales Semiramis-System anmelden und diese Funktion ausführen. Ein Programm würde im Hintergrund den im Abschnitt 3.3.6.1 beschriebenen Ablauf ausführen und die Path-MTU angeben.

Eine andere Möglichkeit wäre es, die Integration in Semiramis ähnlich der *Performance Page* des *Oracle Enterprise Manager* zu gestalten. Auf einer Startseite würden die wichtigsten Systemzustände grafisch dargestellt werden – anhand der in Abschnitt 2.2.6 gezeigten Abbildung der Semiramis-Architektur. Jeder Bereich der Grafik (Datenbanken, System Engine, JVM, Betriebssystem, Netzwerk, usw.) würde farbig hinterlegt werden. Eine grüne Farbe stände für in Ordnung, gelb für einen Warnzustand, rot für einen kritischen Zustand. Diese Zustände könnten aus den in Abschnitt 3.3 aufgezeigten Grenzwerten errechnet werden. Mit Klick auf die jeweiligen Bereiche bekäme man Detailinformationen angezeigt. So würde z.B. ein Klick auf den Bereich Betriebssystem die wichtigsten Betriebssystem Kennzahlen anzeigen. Weiters könnte sich ein Link auf dieser Seite zum Semiramis-Leistungsmonitoring befinden, d.h. der Identifizierung zeitaufwändiger Anwendungen und Berichte. Die Performance-Berichte können ebenfalls von dieser Seite aus aufgerufen werden.

Trotz dieser beiden vorgeschlagenen Integrationen müsste das Datenbank-Tuning immer noch mit den Tools des DBMS durchgeführt werden. Auch Detailanalysen und Abfragen historischer Daten

zu JVM, Datenbank, Hardware und Netzwerk müssten immer noch mit den spezifischen Tools durchgeführt werden. Dies lässt sich aber nicht verhindern, da die spezifischen Monitore sehr umfangreiche Daten liefern, und eine komplette Integration zu aufwändig oder technisch gar nicht möglich wäre.

Die neuen Datenbank-Leistungsmonitore in Semiramis Version 4.2 sind teilweise noch fehlerhaft bzw. existieren Ungereimtheiten. Diese müssen so schnell wie möglich behoben werden. Die leeren Felder im *Systemcockpit* beim Filter *Zeitaufwändige Berichte sortiert nach Summe ausgeben* (siehe Abschnitt 3.3.1.1) werden laut C.I.S. AG durch folgende Umstände verursacht (CIS, 2006e):

- Werden Berichte vom Typ „System“ erstellt (dies sind nicht Standard-Berichte, also Berichte die für den Kunden zusätzlich erstellt werden), dann wird der Name nicht korrekt angezeigt. Dieser Fehler wird wahrscheinlich in der nächsten Semiramis-Auslieferung in einigen Wochen behoben. Es soll dann der Name des Berichtes angezeigt werden. Bis zur Behebung dieses Problems kann mit dieser Anwendung nicht sinnvoll gearbeitet werden, da bei Kundensystemen in der Regel Nicht-Standard Berichte verwendet werden.
- Interaktive ODBC-Zugriffe: Diese kommen z.B. bei einer direkten Ausführung eines Berichtes über Crystal Reports vor (wenn ein sich in Entwicklung befindender Bericht gestestet wird). Auch hier wird kein Name angezeigt. Einträge mit interaktiven ODBC-Zugriffen werden aber kumuliert in einer Zeile dargestellt. In Semiramis 4.3 wird voraussichtlich ein Dummy Bericht mit dem Namen *Interaktiver ODBC-Zugriff* angezeigt werden.

Da der Fehler mit dem Berichtstyp „System“ in dem Bericht *Zeitaufwändige Berichte sortiert nach Summe ausgeben* nicht vorkommt (siehe Abschnitt 3.3.1.2), ist die Verwendung des Berichtes dem *Systemcockpit* vorzuziehen. Die interaktiven ODBC-Zugriffe werden hier als „unbekannt“ angezeigt.

Auch irritiert es, dass für einen Bericht die Anzahl an ODBC-Aufrufen angegeben wird. Besser wäre hier, die Anzahl der Ausführungen eines Berichtes anzugeben. Die Anzahl der ODBC-Aufrufe kann als Zusatzinformation dienen.

Die Verwendung der Performance-Berichte ist der Anzeige im *Systemcockpit* vorzuziehen, da die Daten wesentlich übersichtlicher aufbereitet werden. Auch werden Datenbank-Statements von Unterberichten angezeigt. Die Anzeige im *Systemcockpit* ist für einen schnellen Überblick über das System und zum Auffinden von offensichtlich langsamen Anwendungen gut geeignet. Bis zur Behebung der Fehler im *Systemcockpit* ist zur detaillierteren Analyse das Datei-Leistungsmonitoring immer noch ein wesentlicher Bestandteil der Analyse. Sind alle Fehler behoben, so könnte in den meisten Fällen auf das Datei-Leistungsmonitoring verzichtet werden. Dies würde eine wesentliche Erleichterung darstellen, da Datei-Leistungsmonitoring sehr aufwändig ist.

Diese Beschreibung der Integration in Semiramis wurde an die Analyse von SAP-Systemen (Schneider, 2005) und Kapitel 3 angelehnt. Sie könnte also durchaus auf weitere ERP-Systeme angewendet werden. Dazu müsste ein spezielles ERP-System auf die bereits implementierten

Möglichkeiten der Leistungsanalyse hin untersucht werden, und die Differenzen mit Kapitel 3 dieser Arbeit und (Schneider, 2005) abgeglichen werden.

4.4 Dokumentvorlage

Da die drei oben genannten Lösungsansätze alle einen erheblichen Zeitaufwand verursachen und somit nicht in absehbarer Zeit zur Verfügung stehen werden, wird als kurzfristige Lösung eine Dokumentvorlage erstellt. Diese Dokumentvorlage wird an Kapitel 3 und Abschnitt 2.5 angelehnt und beinhaltet die Abschnitte Zieldefinition, Dokumentation und Validierung der Systemparameter, Analyse der einzelnen Semiramis Komponenten und Präsentieren der Ergebnisse. Diese Vorlage dient dem Performanceanalysten als Anleitung. Zu jedem Gliederungspunkt finden sich Anweisungen, wie vorzugehen ist, bzw. welche Grenzwerte überprüft werden sollen. Dieses Dokument wird vom Analysten mit den Ergebnissen der Analyse ergänzt. Markiert durch „<<…>>“ sind Anweisungen, was der Performanceanalyst im Dokument zu ergänzen hat. *Kursiv* geschrieben sind die jeweiligen Kurzanleitungen. Diese können gelöscht, oder auch zur Veranschaulichung der Analyse im Dokument belassen werden. Speziell die dargestellten Grafiken sollten im Dokument verbleiben. Diese Dokumentvorlage steht als separates Word-Dokument (Format: Microsoft Word 2003) zur Verfügung.

Da diese Vorgehensweise ohnehin in Anlehnung an eine Anleitung zur Analyse von SAP-Systemen erstellt wurde (Schneider, 2005), sollte sie sich auch auf andere ERP-Systeme anwenden lassen. Im Folgenden werden die einzelnen Punkte der Vorgehensweise untersucht:

- Semiramis-Leistungsmonitoring: Dieser Punkt lässt sich am wenigsten verallgemeinern, da jedes ERP-System unterschiedliche Möglichkeiten implementiert hat. Als Mindestanforderung für ein im ERP-System implementiertes Leistungsmonitoring könnte man die Möglichkeit zur Identifizierung von Anwendungen und Berichten mit hoher Ausführungsdauer samt den zugehörigen Datenbankstatements nennen.
- Analyse der JVM: Dies ist ein sehr spezifischer Punkt, da nicht alle ERP-Systeme mit Java implementiert sind. Für ein Java-basiertes ERP-System lässt sich dieser Punkt vollständig anwenden.
- Analyse der Datenbanken: Die Analyse der Datenbanken läuft im Wesentlichen für alle Datenbank-Plattformen – nicht nur für die drei in der Arbeit vorgestellten – nach dem gleichen Schema. Deshalb lässt sich auch dieser Punkt sehr gut auf andere ERP-Systeme anwenden. Speziell die in Abbildung 30 dargestellte Vorgehensweise lässt sich sehr gut Verallgemeinern.
- Hardware-Analyse: Hier wurden die gängigsten Betriebssysteme, die als Datenbank- bzw. Applikation-Server in Frage kommen, betrachtet. Unix-Betriebssysteme bieten ähnliche Befehle wie Linux. Dieser Punkt lässt sich also auch auf andere ERP-Systeme anwenden. Die in Abbildung 40, Abbildung 41 und Abbildung 42 dargestellten Gegenüberstellungen von Performancekennzahlen, Tools, Grenzwerten und Anleitungen können nicht nur zur

Analyse eines Semiramis-Systems, sondern Allgemein zur Analyse von Betriebssystemen herangezogen werden.

- Netzwerk-Analyse: Auch dieser Punkt kann auf andere ERP-Systeme angewendet werden, da nur die Auslastung, die gesendeten bzw. empfangenen Bytes und die Anzahl der Kollisionen bzw. Fehler der Netzwerkinterfaces überwacht werden. Es ist also nicht von Belang über welches Protokoll ein ERP-System kommuniziert. Grundvoraussetzung ist nur die Verwendung von Ethernet und des TCP/IP Protokolls.

5 ZUSAMMENFASSUNG UND AUSBLICK

Zusammenfassend lässt sich eine Analyse des ERP-Systems Semiramis auf die folgenden wesentlichen Punkte vereinheitlichen:

- Identifizierung von Anwendungen bzw. Berichten mit hoher Ausführungszeit und deren zugehörigen Datenbankanweisungen mit dem Semiramis-Leistungsmonitoring.
- Analyse der JVM mit dem in der Version 1.5 mitgelieferten Tool *jconsole*.
- Datenbank-Analyse anhand der in Abschnitt 3.3.4 beschriebenen Vorgehensweise.
- Hardware-Analyse anhand der in Abschnitt 3.3.5 beschriebenen Vorgehensweise.
- Netzwerk-Analyse anhand der in Abschnitt 3.3.6 beschriebenen Vorgehensweise.

Durch die Neuerungen im Leistungsmonitoring der Semiramis Version 4.2 hat sich der erste Punkt der obigen Aufzählung gegenüber früheren Versionen wesentlich vereinfacht. In früheren Versionen war nur ein Datei-Leistungsmonitoring möglich. Jedoch sind speziell bei der Identifizierung von Berichten mit hoher Ausführungszeit noch einige Fehler bzw. Ungereimtheiten vorhanden. Leistungsdaten der darunter liegenden Betriebssysteme und der JVM sind noch nicht in Semiramis integriert. Auch auf Seiten der Datenbank muss immer noch auf Produkte der jeweiligen Hersteller zurückgegriffen werden. Dasselbe gilt für die Analyse des Netzwerkes, hier sind auch keine Möglichkeiten im System vorhanden.

In Abschnitt 4.3 wurde ein Grobkonzept zur Integration dieser Leistungsdaten in Semiramis vorgestellt. Dieses Grobkonzept könnte – unter Berücksichtigung der im anderen ERP-System implementierten Möglichkeiten zur Leistungsanalyse – auch für andere ERP-Systeme als Basis dienen.

Unter Beachtung der in Kapitel 3 vorgestellten Vorgehensweise zur Analyse eines Semiramis-Systems und der erstellten Dokumentvorlage kann die Analyse eines Semiramis-Systems unter vertretbarem Aufwand durchgeführt werden und ein Ergebnis dem Kunden oder einem Entscheidungsträger präsentiert werden.

Die Vorgehensweisen zur Analyse eines Semiramis-Systems – speziell die in Abbildung 21 und Abbildung 30 dargestellten Flussdiagramme – könnten mit relativ wenig Aufwand auf ein anderes ERP-System angepasst werden, da sie an eine Methode zur Analyse von SAP-Systemen angelehnt wurden. Dabei müssten die Architektur, die möglichen Plattformen von Betriebssystem und

Datenbanken, sowie die bereits im anderen ERP-System implementierten Monitoring-Möglichkeiten analysiert werden.

OpenNMS ist in der jetzigen Form nur für Semiramis einsetzbar, könnte aber prinzipiell andere ERP-Systems überwachen, sofern sie SNMP- bzw. JMX-Unterstützung anbieten.

Die in Abbildung 40, Abbildung 41 und Abbildung 42 dargestellten Gegenüberstellungen von Performancekennzahlen, Tools, Grenzwerte und Handlungsempfehlungen können nicht nur zur Analyse eines Semiramis-Systems, sondern allgemein zur Analyse der Hardware herangezogen werden.

Für das ERP-System Semiramis ist weiteres Ausbaupotential vorhanden, vergleicht man es mit den Monitoring-Möglichkeiten, die ein SAP-System bietet. Als Ausblick in die nähere Zukunft können die folgenden Punkte festgehalten werden:

- Die in Kapitel 3 vorgestellte Vorgehensweise zur Analyse eines Semiramis-Systems und die erstellte Dokumentvorlage dienen bis auf weiteres als Leitfaden zur Analyse eines Semiramis-Systems für KTW-Mitarbeiter, aber auch für KTW-Kunden. Die Dokumentvorlage bzw. diese Arbeit könnte aber auch zur Einschulung neuer Mitarbeiter verwendet werden.
- Bei der C.I.S. AG sollte angefragt werden, ob Interesse besteht, weitere Leistungsdaten in näherer Zukunft in Semiramis zu integrieren. Bei Interesse können die in Kapitel 3 vorgestellten Kennzahlen und Grenzwerte bzw. das in Abschnitt 4.3 beschriebene Grobkonzept als Grundlage für ein Software-Konzept dienen. Die neuen Datenbank-Leistungsmonitore in Semiramis Version 4.2 sind teilweise noch fehlerhaft bzw. existieren Ungereimtheiten. Diese müssen so schnell wie möglich behoben werden. Durch eine erweiterte Integration könnte sich ein Wettbewerbsvorteil für Semiramis ergeben, wenn dies entsprechend vermarktet wird, speziell bei zukünftigen Systemverantwortlichen.
- Fällt die Reaktion der C.I.S. AG negativ aus, könnte *OpenNMS* entsprechend erweitert werden. Hier würden wiederum die in der Arbeit vorgestellten Kennzahlen und Grenzwerte als Grundlage für ein Software-Konzept dienen.
- Ein Enterprise Management Tool auf Semiramis maß zu schneidern ist momentan noch nicht anzuraten, da sich der Marktanteil von Semiramis dafür noch sehr stark erhöhen müsste. Jedoch sind solche maßgeschneiderten Tools für ERP-Systeme wie SAP, Peoplesoft oder Siebel am Markt vorhanden. Dies könnte einen Wettbewerbsnachteil für Semiramis bedeuten.

Da eine Integration von weiteren Leistungsdaten die Wettbewerbschancen von Semiramis erhöhen würden, und auch eine wesentliche Zeitersparnis für Systemverantwortliche beinhalten würde, wäre dies auf jeden Fall anzuraten.

LITERATURVERZEICHNIS

Anderson, 2005	R. Anderson: Monitoring Linux with native tools. In 30th Annual Internal Conference of The Computer Measurement Group Inc., December 5-10, 2004. – Las Vegas: CMG, 2005
Anderson, 2005a	G. Anderson; M. Mißbach: Projekthandbuch Last-Testing und Performance Tuning. – Bonn: SAP Press, 2005.
Bartley, 2005	R. Bartley et. al.: IBM @server iSeries Performance Management Tools. – Armonk: IBM Corp., 2005.
Bauder, 2006	I. Bauder: Microsoft SQL Server 2005 für Administratoren. – München, Wien: Hanser, 2006.
Bedernjak, 2005	M. Bedernjak et. al.: IBM @server i5 and iSeries System Handbook IBM i5/OS Version 5 Release 3. – Armonk: IBM Corp., 2005.
Bedoya, 2004	H. Bedoya et. al.: DB2 Universal Database for iSeries Administration. – Armonk: IBM Corp., 2004.
Böhmer, 2005	W. Böhmer: VPN: Virtual Private Networks, 2. überarbeitete Auflage. – München: Hanser, 2005.
Boswell, 2005	W. Boswell: Windows Server 2003 für Insider. – München: Markt und Technik Verlag, 2005.
Bragg, 2004	R. Bragg et. al.: Network Security: The Complete Reference. – Emeryville: McGraw-Hill/Osborne, 2004.
Brause, 2001	R. Brause: Betriebssysteme – Grundlagen und Konzepte, 2. Auflage. – Berlin: Springer, 2001.
Chan, 2005	I. Chan et. al.: Oracle® Database Performance Tuning Guide 10g Release 2 (10.2). – Redwood City: Oracle Corporation, 2005.
Choi, 2005	P. Choi et. al.: Oracle® Enterprise Manager Concepts 10g Release 2 (10.2). – Redwood City: Oracle Corporation, 2005.
Chung, 2004	M. Chung: Using Jconsole to Monitor Applications, 2004. – URL: http://java.sun.com/j2se/1.5.0/docs/guide/management/jconsole.html, [Stand: 13.03.2006].
CIS, 2004	o. V.: Technische Dokumentation: Semiramis Stabilitätstests und Sizing mit OpenSTA. – Hannover: C.I.S. Cross Industrie Software® AG, 2004.
CIS, 2004a	o. V.: Semiramis Produktinformationen. Technische Lösungen. – Hannover: C.I.S. Cross Industrie Software® AG, 2004. (internes Dokument).
CIS, 2004b	o. V.: Technische Dokumentation: Systemlandschaft einrichten. – Hannover:

C.I.S. Cross Industrie Software® AG, 2004.

CIS, 2005 o. V.: Technische Dokumentation: Datenbankabfragen anzeigen. – Hannover: C.I.S. Cross Industrie Software® AG, 2005.

CIS, 2005a o. V.: Technische Dokumentation: Leistungsoptimierung. – Hannover: C.I.S. Cross Industrie Software® AG, 2005.

CIS, 2005b o. V.: Technische Dokumentation: Checkliste: System-Konfiguration. – Hannover: C.I.S. Cross Industrie Software® AG, 2005.

CIS, 2006 o. V.: Technische Dokumentation: Leistungsmonitore. – Hannover: C.I.S. Cross Industrie Software® AG, 2006.

CIS, 2006a o. V.: Technische Dokumentation: Systemcockpit: Typ „System“. – Hannover: C.I.S. Cross Industrie Software® AG, 2006.

CIS, 2006b o. V.: Technische Dokumentation: Systemcockpit: Typ „Application-Server“. – Hannover: C.I.S. Cross Industrie Software® AG, 2006.

CIS, 2006c o. V.: Technische Dokumentation: Leistungsinformationen protokollieren und auswerten. – Hannover: C.I.S. Cross Industrie Software® AG, 2006.

CIS, 2006d o. V.: Technische Dokumentation: JVM-Einstellungen. – Hannover: C.I.S. Cross Industrie Software® AG, 2006.

CIS, 2006e R. Gohla.: Email Korrespondenz mit Hr. Gohla von der Firma C.I.S. – Hannover: C.I.S. Cross Industrie Software® AG, 2006.

Conrads, 2004 D. Conrads: Telekommunikation: Grundlagen, Verfahren, Netze, 5. Auflage. – Wiesbaden: Friedr. Vieweg & Sohn Verlag, 2004.

Crystal, 2006 o. V.: Homepage der Firma Business Objects. – URL: http://www.businessobjects.com/products/reporting/crystalreports/default.asp , [Stand 10.06.06].

Date, 2004 Chris J. Date: An Introduction to Database Systems, 7th Edition. – Upper Saddle River: Addison Wesley, 2004.

Delany, 2001 K. Delany et al.: Inside Microsoft SQL Server 2000. – Redmond: Microsoft Press, 2001.

Fortier, 2003 Paul J. Fortier et. al.: Computer Systems Performance Evaluation and Prediction. – Burlington: Digital Press, 2003.

Gartner, 2000 B. Bond, et. al.: Research Note: ERP Is Dead – Long Live ERP II. – o.O.: Gartner Group, 2000.

Härder, 2001 T. Härder et. al.: Datenbanksysteme: Konzepte und Techniken der Implementierung. – Berlin: Springer, 2001.

Hoermann, 2006 M. Hoermann: Der Ausführungsplan – das unbekannte Wesen. - Paderborn: ORDIX AG, 2006.

HP, 2006 o. V.: System- und Applikationsmanagement mit HP OpenView Operations. – URL: http://h40047.www4.hp.com/software/openview/systemmanagement.php, [Stand 19.05.06].

Hunt, 2003 C. Hunt: TCP/IP Netzwerk Administration, Deutsche Ausgabe der 3. Auflage. – Köln: O'Reilly, 2003.

IBM, 2005 o. V.: Online Hilfe des iSeries Navigator V5R4M0. – Armonk: IBM Corp., 2005.

IEEE, 2006 o. V.: IEEE 802.3-REVam-2005 Standard: IEEE Standard for Information technology – Telecommunications and information exchange between systems – – Local and metropolitan area networks – specific requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. – New York: IEEE, 2006.

Jain, 1991 Jain, Raj: The Art of Computer Systeme Performance Analysis: techniques for experimental design, measurement, simulation and modelling. – New York: Wiley, 1991.

Johnson, 2005 P. Johnson: Java Garbage Collection Statistical Analysis 201. In 30th Annual Internal Conference of The Computer Measurement Group Inc., December 5-10, 2004. – Las Vegas: CMG, 2005

Jones, 2002 D. Jones: SQL Server Performance Optimization. – o.O: Real Time Publishers, 2002.

KTW, 2006 o. V.: Homepage der Firma KTW. – URL: http://www.ktw.com, [Stand 17.03.06].

KTW, 2006a R. Koidl.: Seminarunterlagen: Releasevorstellung i5/OS V5R4M0. – Kirchbichl: KTW Software & Consulting GmbH, 2006

Kurose, 2002 James F. Kurose; Keith W. Ross: Computernetze: Ein Top-Down-Ansatz mit Schwerpunkt Internet. – München: Pearson, 2002.

Linux, 2006 o. V.: sar(1) – Linux man page. – URL: http://www.die.net/doc/linux/man/man1/sar.1.html , [Stand 04.05.06].

Märtin, 2003 C. Märtin: Einführung in die Rechnerarchitektur: Prozessoren und Systeme. – Leipzig: Hanser, 2003.

Menascé, 2002 D. A. Menascé; V.A.F. Almeida: Capacity Planning for Web Services: Metrics, Models, and Methods. – Upper Saddle River: Prentice Hall, 2002.

Menascé, 2004 D. A. Menascé; V.A.F. Almeida et al: Performance by Design: Computer Capacity Planning by Example. – Upper Saddle River: Prentice Hall, 2004.

Mertens, 2001 P. Mertens.: Integrierte Informationsverarbeitung #Bd. 1, 13. Auflage. – Wiesbaden: Gabler, 2001.

Microsoft, o. V.: Chapter 27 – Overview of Performance Monitoring. – URL:

2006 http://www.microsoft.com/technet/prodtechnol/Windows2000Pro/reskit/part6/proch27.mspx, [Stand 15.03.06].

MOM, 2006 o. V.: Microsoft Operations Manager Homepage. – URL: http://www.microsoft.com/mom/default.mspx, [Stand 19.05.06].

NetIQ, 2006 o. V.: Homepage der Firma NetIQ. – URL: http://www.netiq.com/products/am/default.asp, [Stand 19.05.06].

Neumann, 2002 H. Neumann.: Das Lexikon der Internetpioniere. – Berlin: Schwarzkopf & Schwarzkopf, 2002.

OpenSTA, 2006 o. V.: OpenSTA Documentation. – URL: http://www.opensta.org/docs/, [Stand 10.06.06].

Oracle, 2006 o. V.: Excilla Dot Net. – URL: http://www.exzilla.net/docs/architecture/oracle-architecture.gif, [Stand 19.05.06].

Panko, 2004 Raymond R. Panko: Corporate Computer and Network Security. – Upper Saddle River: Pearson, 2004.

Schneider, 2005 T. Schneider: SAP-Performanceoptimierung: Analyse und Tuning von SAP-Systemen, 4. aktualisierte und erweiterte Auflage. – Bonn: SAP Press, 2005.

Schwartz, 2005 J. A. Schwartz: Understanding and Interpreting SQL Server Performance Counters. In 30th Annual Internal Conference of The Computer Measurement Group Inc., December 5-10, 2004. – Las Vegas: CMG, 2005

Schwarzer, 2004 B. Schwarzer et al.: Wirtschaftsinformatik, 3. Auflage. – Stuttgart: Schäffer-Poeschel, 2004.

Segue, 2006 o. V.: Homepage der Firma Segue Software. – URL: http://www.segue.com/products/monitoring/silkcentral-performance-manager.asp, [Stand 19.05.06].

Shirazi, 2003 J. Shirazi: Java Performance Tuning, 2nd Edition. – Sebastopol : O'Reilly, 2003.

Silberschatz, 2005 A. Silberschatz et al.: Operating System Concepts, 7th Edition. – New York: Wiley, 2005.

Tanenbaum, 2002 Andrew S. Tanenbaum: Moderne Betriebssysteme, 2. überarbeitete Auflage. – München: Pearson, 2002.

Tanenbaum, 2003 Andrew S. Tanenbaum: Computernetzwerke, 4. überarbeitete Auflage. – München: Pearson, 2003.

Tivoli, 2006 o. V.: IBM Tivoli Monitoring for Applications. – URL: http://www-306.ibm.com/software/tivoli/products/monitor-apps/, [Stand 19.05.06].

Türker, 2005 C. Türker; G. Saake: Objektrelationale Datenbanken: ein Lehrbuch. – Heidelberg: dpunkt, 2005.

Vossen, 2000 G. Vossen: Datenmodelle, Datenbanksprachen und

Datenbankmanagementsysteme, 4. Auflage. – München: Oldenburg, 2000.

Whalen, 2005 E. Whalen et. al.: Oracle Database 10g: Linux Administration. – Emeryville: Oracle Press, 2005.

Wielsch, 1999 M. Wielsch et. al.: LINUX intern: Technik, Administration und Programmierung. – Düsseldorf: Data Becker, 1999.

Wiki, 2006 o. V.: Wikipedia – System i5. – URL: http://de.wikipedia.org/wiki/AS/400, [Stand 10.06.06].

Wurzrainer, 2006 K. Wurzrainer: Monitoring für komplexe Java-Anwendungen mit Java 5.0. – Rosenheim: Diplomarbeit FH Rosenheim, 2006.

A ANHANG

A.1 Dokumentvorlage zur Analyse eines Semiramis-Systems

Leistungsanalyse

<<Datum>>

<<Fügen Sie oben das Firmenlogo ein>>

für:

<<Kundenname>>
<<Adresse>>
<<PLZ – Ort>>

A.1.1 Zieldefinition

<<Beschreiben Sie hier die Ziele der Analyse. Auch der Zeitraum der Analyse sollte angegeben werden>>

Es ist wichtig, vom Kunden eine präzise Beschreibung des momentanen und gewünschten Systemverhaltens zu bekommen. Beispielsweise sollte angegeben werden, zu welchen Zeiträumen inakzeptable Antwortzeiten vorkommen, oder welche Anwendungen bzw. Berichte beteiligt sind. Ein Kunde sollte auch seine IT-Infrastruktur dahingehend prüfen, ob andere Anwendungen (z.B. Backups) das Semiramis System beeinflussen. Befasst sich ein Kunde mit seinem System kann es sein, dass die Ursache selber gefunden wird. Beispielsweise wird eine große Anzahl von Belegen – wie etwa Auftragsbestätigungen – zu einem bestimmten Zeitpunkt gedruckt und das System wird dadurch langsam. Eine Verbesserung der Performance kann hier leicht durch Verlagern der Ausdrucke auf den Abend oder die Mittagspause erreicht werden. Aber es kann auch der umgekehrte Fall eintreten. Beispielsweise verlangt ein Kunde Antwortzeiten, die nicht realistisch sind. Hier muss dem Kunden klar gemacht werden, dass solche Antwortzeiten nicht möglich sind. Beispielsweise wurde ein altes textbasiertes ERP-System eines Kunden durch Semiramis ersetzt. Der Kunde beklagt sich, dass im neuen System alles viel länger dauert. Die Erklärung ist hier, dass in einem graphischen System Aktionen länger dauern können als in textbasierten Systemen. Das Beschreiben des Systemverhaltens ist aber nicht immer einfach. Ist eine Transaktion sehr langsam, kann das weit reichende Auswirkungen auf das gesamte System haben. So kann es vorkommen, dass die Ursache eines Performanceproblems anderswo liegt, als dies vom Benutzer wahrgenommen wird.

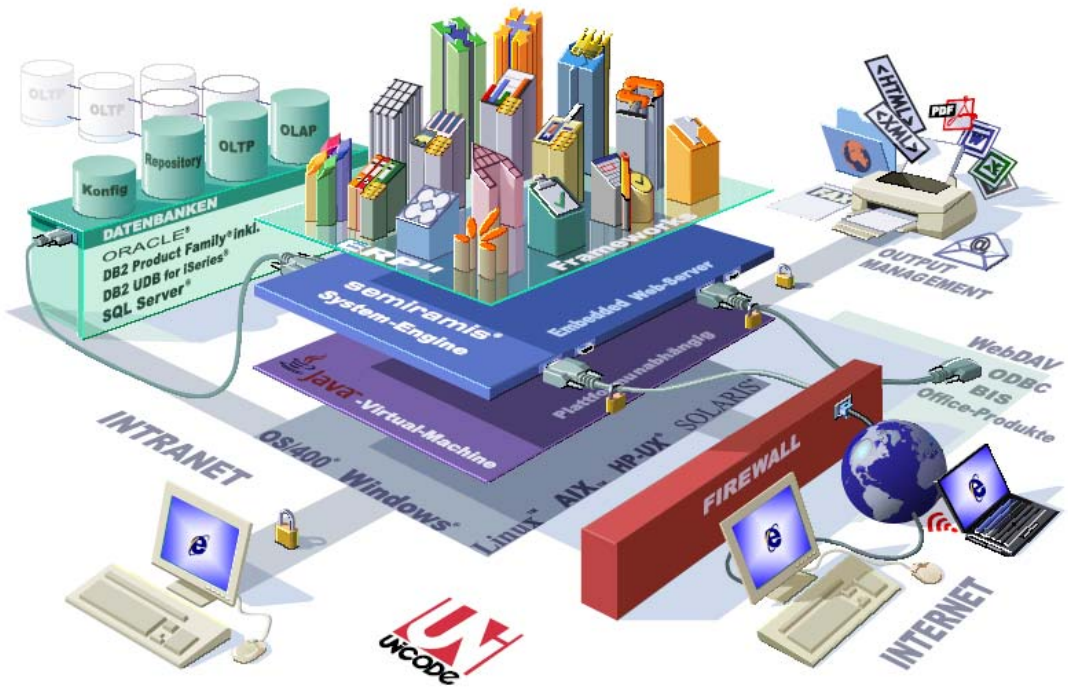
A.1.2 Dokumentation und Validierung der Systemparameter

<<Stellen Sie hier die Architektur der Semiramis Systemlandschaft grafisch dar und Dokumentieren sie die Systemeinstellungen. Zum Überblick kann auch die Grafik der Semiramis Architektur verwendet werden, die sich weiter unten befindet>>

Bevor mit dem Monitoring begonnen wird, muss die Architektur des Semiramis Systems beschrieben werden. Dazu zählen die jeweiligen Betriebssystem- und Datenbank-Plattformen, sowie die Anzahl der SAS und SOM, und deren Verteilung auf unterschiedliche Server. Hier bietet es sich an, eine Skizze der Systemlandschaft zu erstellen.

Anschließend werden die Systemeinstellungen geprüft. Anleitungen dazu finden sich in der Semiramis Hilfe (Technische Dokumentation: Systemlandschaft einrichten, Checkliste: System Konfiguration, JVM-Einstellungen, Leistungsoptimierung). Die Systemeinstellungen eines Semiramis Systems können im Systemcockpit, aber auch anhand von Konfigurationsdateien

eingestellt werden. Es sollten aber auch die Konfigurationen der Betriebssysteme, Datenbanken und JVM's mit den jeweiligen Verwaltungswerkzeugen geprüft werden. Auch die Netzwerk-Infrastruktur ist zu prüfen. Möglicherweise wird bei der Kontrolle der Systemeinstellungen schon eine offensichtliche Fehleinstellung gefunden, die das Performanceproblem behebt. Ein Beispiel dazu wäre eine zu hoch eingestellte Anzahl von möglichen Threads in der Job-Queue. Dadurch können zu viele Hintergrundjobs parallel laufen, und das Gesamtsystem wird so stark belastet, dass für andere Anwendungen keine Ressourcen mehr zur Verfügung stehen.



A.1.3 Analyse der einzelnen Semiramis Komponenten

<<Führen Sie nun die Analyse der einzelnen Semiramis Komponenten durch. Eine Anleitung zur Analyse findet sich in Abschnitt 3.3. Beschreiben Sie hier für den Kunden kurz, wie Sie bei der Analyse vorgegangen sind. Welche Fehler haben Sie gefunden, bzw. wie können diese gelöst werden? Strukturieren Sie das Dokument wie folgt:

- Monitoren des Systems und Analysieren der Resultate: Beschreiben Sie hier, wie und wo sie die Daten gesammelt und analysiert haben, und welche Fehler sie gefunden haben.
- Erstellen von Hypothesen: Stellen Sie hier Hypothesen dar, die ihrer Meinung nach die Ursachen für die vorher gefunden Fehler sein könnten. Wenn es mehrere Ursachen gibt, ist es sinnvoll, diese Schritt für Schritt zu behandeln.

- Implementieren der Lösungen: Stellen Sie hier die Lösungen für die in der Hypothese beschriebenen Fehler dar. Nach Möglichkeit implementieren Sie die Lösungen gleich. Wichtig ist es, immer eine Lösung nach der anderen zu implementieren.
- Test und Überwachung der Lösungen: Überprüfen Sie, ob die Änderungen eine Verbesserung gebracht haben>>

Nachfolgend eine kurze Beschreibung der Möglichkeiten, die zur Verfügung stehen:

Identifizierung von Anwendungen bzw. Berichten mit hoher Ausführungszeit und deren zugehörigen Datenbankanweisungen mit dem Semiramis Leistungsmonitoring:

Das Ziel einer Performancenalyse mit dem Semiramis Leistungsmonitoring sollte sein, Aktionen, die eine lange Ausführungszeit haben, zu finden und zu optimieren. Es kann hier keine pauschale Aussage über akzeptable Antwortzeiten getroffen werden. Für Semiramis gelten z.B. für die Ausführungszeit von Berichten oder Anwendungen Antwortzeiten von 0-5 Sekunden als akzeptabel (siehe Technische Dokumentation: Leistungsinformationen protokollieren und auswerten). Dies ist jedoch immer sehr kundenspezifisch zu sehen. So kann ein Kunde mit Antwortzeiten von 5 Sekunden zufrieden sein, ein Anderer findet dies als störend. Für Aktionen, die oft am Tag ausgeführt werden, werden lange Antwortzeiten eher als störend empfunden als für Anwendungen, die nur einmal pro Woche ausgeführt werden.

Semiramis bietet 3 Möglichkeiten zur Performanceanalyse: Datenbank-Leistungsmonitore und deren Auswertungsmöglichkeiten im Systemcockpit, Performance-Berichte (beides neu in Version 4.2) und das Datei-Leistungsmonitoring. Siehe dazu Abschnitt 3.3.1 und das Flussdiagramm weiter unten.

Analyse der JVM:

Die Analyse der JVM kann mit dem in der Version 1.5 gelieferten Programm jconsole durchgeführt werden.

Siehe dazu http://java.sun.com/j2se/1.5.0/docs/guide/management/jconsole.html.

Analyse der Datenbanken:

Für die Analyse der Datenbanken siehe Abschnitt 3.3.4 und das Flussdiagramm weiter unten. Folgende Vorgehensweisen empfehlen sich für Datenbanken:

- *Analyse der Datenbankpuffer: Kontrolle der Hit-Ratios der Datenbankbuffer. Diese sollten knapp bei 100% liegen. Es sollte genügend freier Speicherplatz in den Datenbankbuffern vorhanden sein. Kontrolle der Schreib- und Lesevorgänge auf die Datenträger. Wenn diese konstant hoch sind, und die Hit-Ratios konstant niedrig, signalisiert das ein Speicherproblem.*

- *Identifizierung teurer SQL-Anweisungen: Die Identfizierung von teuren SQL-Anweisungen läuft auf allen 3 Datenbank-Plattformen nach dem gleichen Schema:*
  - *Zuerst müssen die SQL-Statements gesucht werden, die eine hohe Ausführungszeit bzw. einen hohen Anteil an der Gesamtaktivität des Systems haben. Oracle gibt den prozentuellen Anteil der Top 10 SQL-Anweisungen an der Gesamtaktivität des Systems an und sortiert diese gleich in absteigender Reihenfolge. Beim SQL-Server und bei der DB2 UDB for iSeries wird die Ausführungszeit pro Statement angegeben. Um Statements mit hoher Ausführungszeit zu finden, muss nach der Ausführungszeit in absteigender Reihenfolge sortiert werden. Zu beachten ist, dass im Unterschied zum Semiramis-Leistungsmonitoring hier jedes einzelne Statement aufgelistet ist, und keine Aggregation stattfindet. Um Aggregationen durchzuführen, müssen die Daten in einer Tabelle gespeichert, und per SQL ausgewertet werden. So lassen sich die Anzahl der Ausführungen, die summierte Ausführungszeit, oder die durchschnittliche Ausführungszeit eines Statements anzeigen. Meistens wurde jedoch das zu optimierende Statement bereits durch das Semiramis-Leistungsmonitoring gefunden, so muss nur noch dieses Statement gefunden werden. Dieses Statement zu finden, ist aber oft nicht so einfach, da sich die Statement-ID's und auch die Syntax der Statements zwischen Semiramis und dem DBMS unterscheiden. Die beste Möglichkeit ein Statement zu suchen, das im Semiramis-Leistungsmonitoring gefunden wurde, ist es, die Anwendung welches das Statement verwendet (wird im Leistungsmonitor angegeben) erneut auszuführen. Lässt man nun gleichzeitig eine Ablaufverfolgung auf der Datenbank laufen, sollte dieses Statement rasch gefunden werden.*
  - *Analyse des Zugriffsplans und Optimieren des Statements, beispielsweise durch Indizes.*
  - *Testen, ob die Optimierung eine Verbesserung gebracht hat. Im Fall eines Index, Überprüfung ob dieser verwendet wird. Die Ausführungszeit des optimierten Statements sollte mit der des alten Statements verglichen werden. Der aussagekräftigste Test ist es aber, das System mit realen Benutzern zu testen. Eine Optimierung hat nur dann etwas gebracht, wenn dies von einem Benutzer spürbar wahrgenommen wird.*
  - *Wenn der Index eine Verbesserung gebracht hat, muss dieser in Semiramis in der Anwendung Individuelle Indizes angelegt werden, da er ansonsten bei der nächsten Software-Aktualisierung nicht mehr vorhanden ist.*
- *Identifizierung von Schreib/Lese-Problemen: Stellt man fest, dass Lese- und Schreibvorgänge sehr langsam sind, oder I/O-Warteschlangen sehr lang sind, dann kann es Vorteile bringen, häufig verwendete Dateien auf unterschiedliche Datenträger zu legen. Dazu muß eine Analyse der Verteilung der I/O-Last durchgeführt werden. Stellt man fest, dass beispielsweise 2 Datenbankfiles häufig gelesen oder geschrieben werden, die sich auf*

einem Datenträger befinden, so kann die Verteilung dieser beiden Dateien auf unterschiedliche Platten eine Performanceverbesserung bringen. Im Oracle Enterprise Manager kann die Verteilung der I/O-Last unter dem Punkt File IO Stats eingesehen werden. Beim SQL-Server konnte keine solche Anzeige ermittelt werden. Für die DB2 UDB for iSeries ist die Analyse der I/O-Last nicht relevant.

- *Identifizierung von Sperrproblemen: Sperren beeinflussen die Performance eines DBMS negativ, deshalb sollten sie so selten wie möglich auftreten. Treten Sperren über längere Zeiträume oder Deadlocks auf (d.h. sie konnten vom DBMS nicht beseitigt werden), können folgende Maßnahmen zur Beseitigung der Sperre oder des Deadlocks führen: Wenn die Sperre von einem Programm gehalten wird, das offensichtlich nicht von Semiramis stammt (z.B. eine zeitraubende externe Abfrage mit Microsoft Query, oder ein Prozess, der nicht ordnungsgemäß beendet wurde), kann es beendet werden. Wird eine Sperre von einem Programm über mehrere Minuten gehalten, so ist Rücksprache mit dem Benutzer zu nehmen, damit dieser die Anwendung beendet, die er gerade bearbeitet. Trifft keiner der oben genannten Punkte zu, so kann es durch generelle Performanceprobleme der Datenbank vorkommen, das Sperren lange gehalten werden. In diesem Fall ist zuerst das Performanceproblem zu beheben. Eine hohe Anzahl an gehaltenen Sperren kann aber auch durch schlecht programmierte Anwendungen hervorgerufen werden.*

Hardware-Analyse:

Für die Analyse der Hardware siehe Abschnitt 3.3.5 und die 3 weiter unten dargestellten Tabellen.

Netzwerk-Analyse:

Für die Analyse des Netzwerkes siehe Abschnitt 3.3.6.

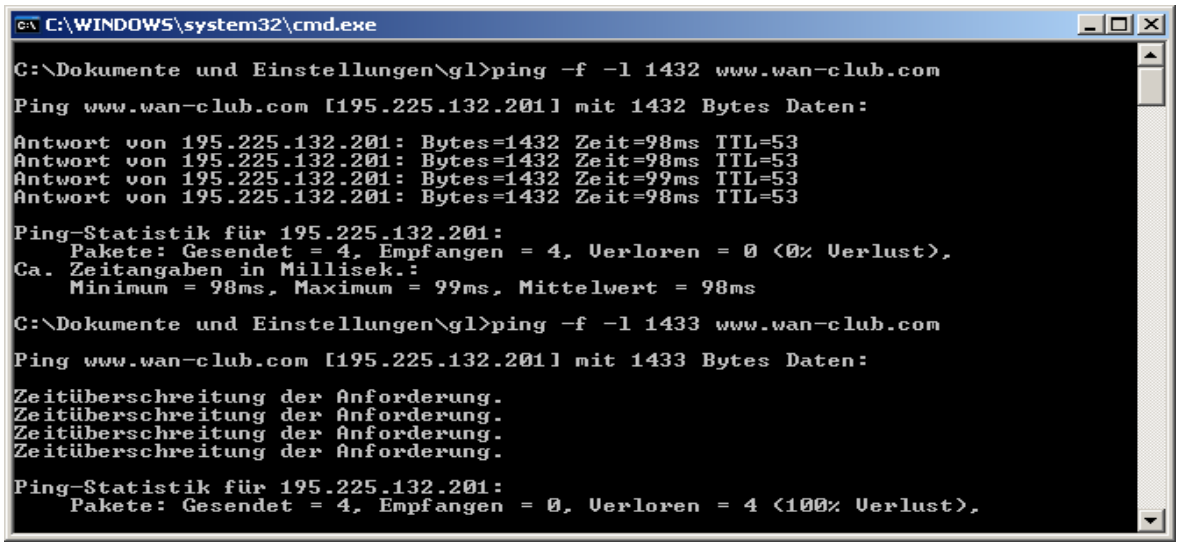
Wichtige Kommandozeilenbefehle zum Monitoring des Netzwerkes unter Windows sind:

- *ping: Dient üblicherweise zur Überprüfung, ob ein Netzwerkgerät erreichbar ist oder nicht. Der Befehl ping kann aber auch für weitere Diagnosen herangezogen werden, beispielsweise für die Ermittlung der Path-MTU.*
- *tracert: zeichnet den Pfad eines Requests auf. Es werden alle durchlaufenden Subnets angegeben und auch die Zeit, die der Request benötigt, das jeweilige Subnet zu durchlaufen. So kann ein Subnet identifiziert werden das Probleme macht.*

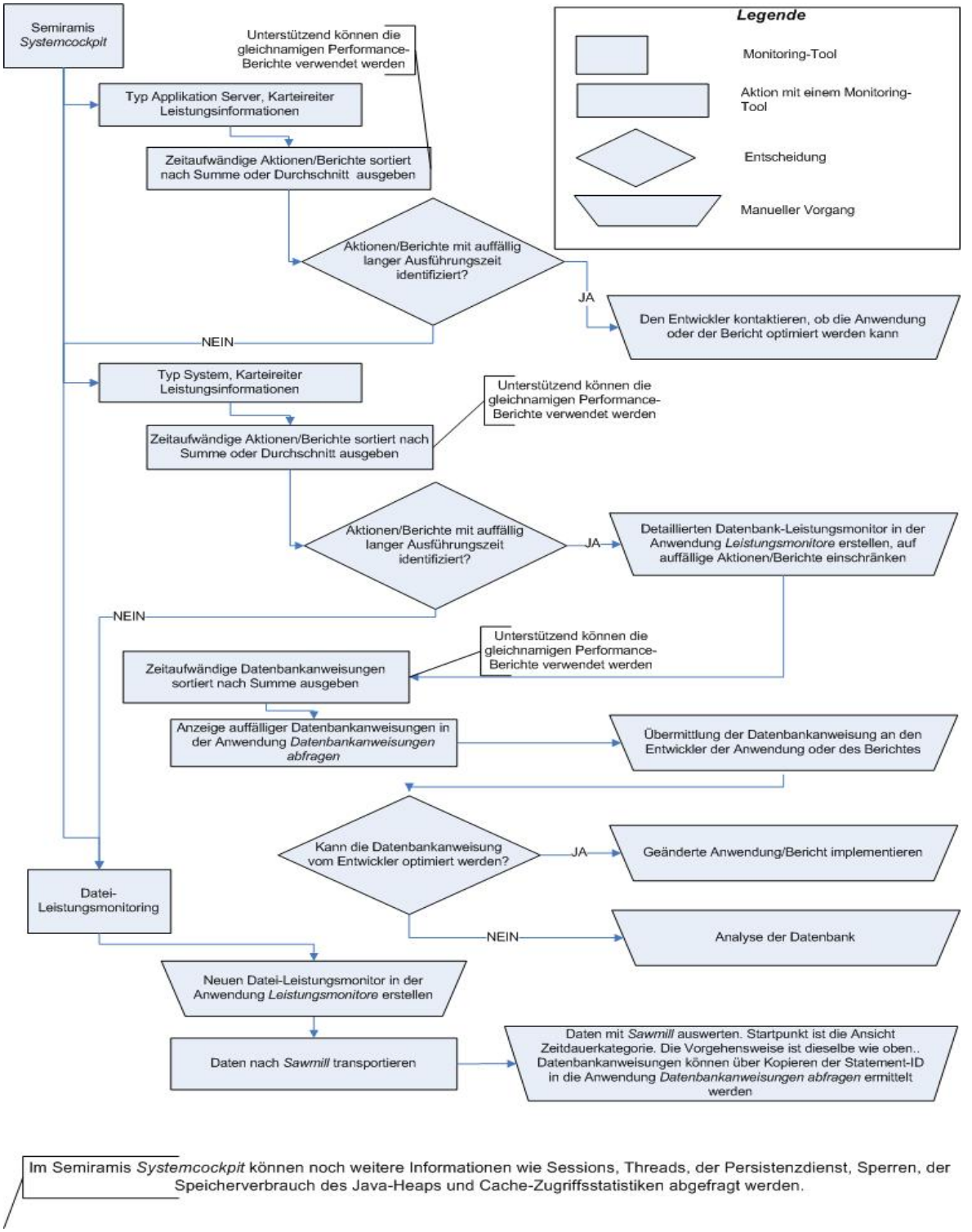
Ermittlung der Path-MTU

Um die Path-MTU - die bei Semiramis speziell für VPN-Verbindungen von Außenstellen interessant ist – zu ermitteln, kann der Befehl ping verwendet werden. Es muss das dont' fragment Flag gesetzt werden, und die die Länge des Paketes solange erhöht werden bis der Zielhost nicht mehr erreichbar ist. Der letzte Wert der Paketlänge bei dem der Ziel-Host noch erreichbar ist (das

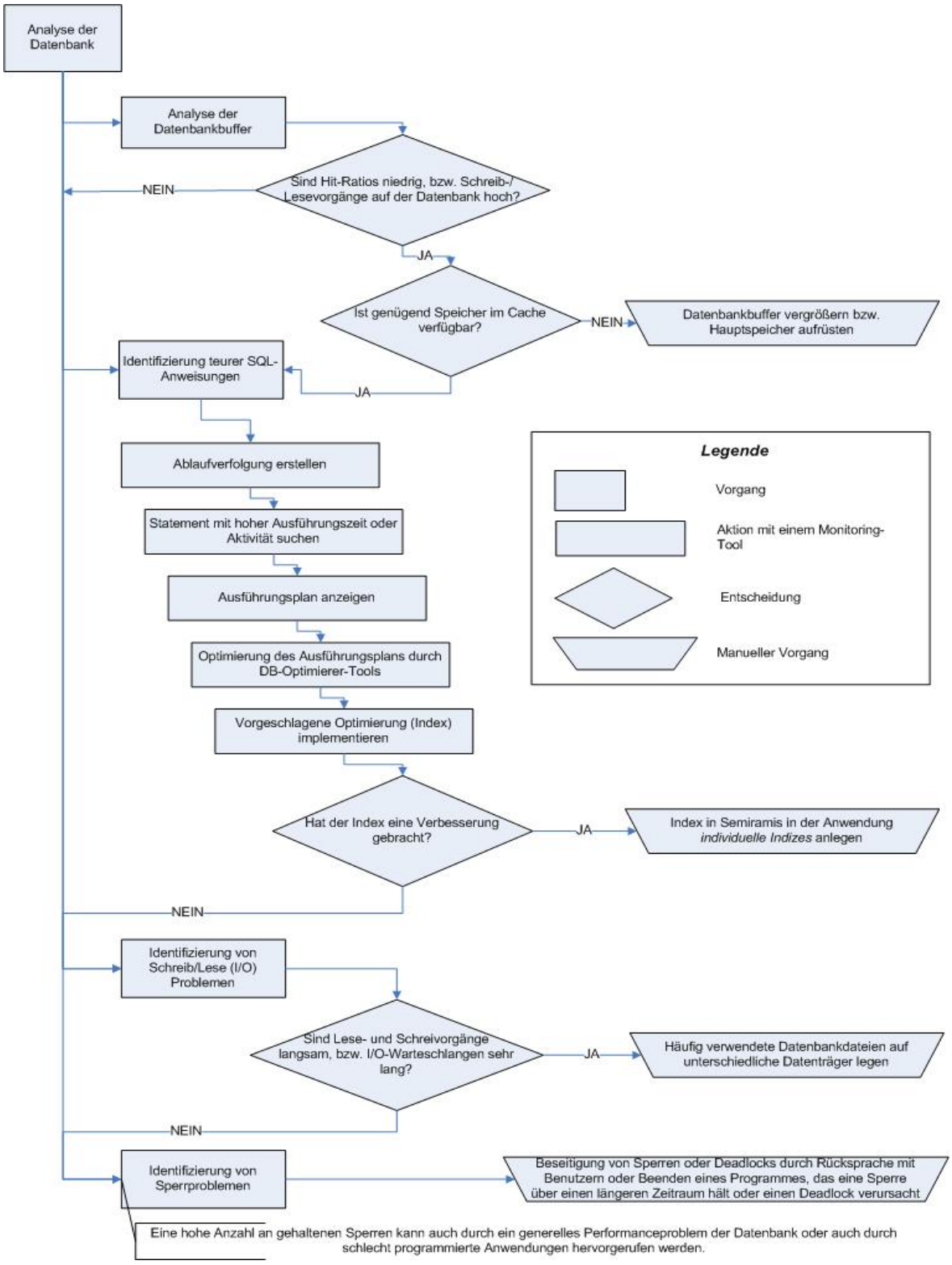
dont' fragment Flag bewirkt, dass ein Paket das fragmentiert werden müsste, nicht mehr gesendet wird), entspricht der Path-MTU. Im Beispiel der folgenden Abbildung ist die Path-MTU 1432. Die Option –f setzt das don't fragment Flag, die Option –l gibt die Paketgröße an. Die Implementierungen des ping Kommandos sind von Betriebssystem zu Betriebssystem unterschiedlich.



Vorgehensweise zur Identifizierung von Anwendungen bzw. Berichten mit hoher Ausführungszeit und deren zugehörigen Datenbankanweisungen mit dem Semiramis-Leistungsmonitoring:



Vorgehensweise zur Datenbank-Analyse



Gegenüberstellung wichtiger Kennzahlen, Tools, Grenzwerte und Handlungsempfehlungen für die Betriebssysteme Windows, Linux und i5/OS.

CPU

Kennzahl	**Windows: *perfmon*, Task Manager**	**Linux: *top*, sar, vmstat**	**i5/OS: *iSeries Navigator*, wrkactjob, wrksyssts**	**Grenzwerte**	**Kommentare/ Handlungsempfehlungen**
CPU Auslastung	Prozessorzeit (%) Task Manager: CPU-Zeit pro Prozess	CPU states, %CPU pro Job sar: sar -u Ausgabe	Systemstatus: Feld CPU Belastung abgelaufen Systemüberwachung: CPU-Auslastung (durchschnittlich) wrkactjob bzw. aktive Jobs (CPU pro Job)	Windows: < 85% Linux: im Mittel zwischen 0 und 50% i5/OS: < 90%, CPU pro Job <40%	Finden und Analysieren der Prozesse, die einen hohen Anteil an der Prozessorzeit haben. Aufrüsten auf einen schnelleren Prozessor, oder Hinzufügen eines weiteren Prozessors.
Interrupts	Interrupts/s	vmstat: Sektion system, Spalte in	wrksyssts, Spalte Act-Wait	abhängig vom Prozessor. Linux, Windows: Für aktuelle CPU's < 1500 i5/OS: keine generelle Aussage zu treffen, da Hardware über IOP verwaltet wird	Eine dramatische Erhöhung dieser Kennzahl ohne korrespondierende Erhöhung der Systemaktivität lässt auf ein Hardwareproblem schließen. Identifizierung des Netzwerk-Adapters oder des Festplattencontrollers, der dieses Problem verursacht.
CPU-Warte-schlangen-länge	Warteschlangen-länge	load averages sar: sar -q Ausabe	Ablaufsteuerung /Jobwarteschlangen (Achtung: dies sind keine CPU Wartschlangen, sondern Jobwarteschlangen pro Subsystem)	Linux, Windows <4 i5/OS: Kontrolle, ob ein Job in einer weiteren Warteschlange mit niedriger Priorität vorhanden ist	Ein Erreichen dieses Grenzwertes kann einen Prozessor Flaschenhals signalisieren. Diese Kennzahl muss zur genaueren Aussage über einen längeren Zeitraum beobachtet werden.

Speicher

Kennzahl	Windows: *perfmon*, Task Manager	Linux: *vmstat*, sar, top	i5/OS: *iSeries Navigator*, wrksyssts	Grenzwerte	Kommentare/ Handlungsempfehlungen
Belegung der Auslagerungs-datei bzw. des swap oder temporären Speichers	Belegung (%)	Memory Sektion, Spalte swpd, top: Swap Zeile sar: sar -B, Spalte kbswpused	Systemstatus, Reiter Plattenpeicher-platz, Sektion belegter temporärer Speicher wrksyssts: Current unprotect used, Maximum unprotect	Windows: < 70% Linux: <98% bei i5/OS nicht relevant, der temporäre Speicher kann einige hundert GB groß sein. Wenn der temporäre Speicher sehr stark über die Zeit anwächst, signalisiert dies ein Problem ("Speicherfresser-Anwendung").	Dieser Wert ist immer in Zusammenhang mit dem verfügbaren physikalischen Speicher zu sehen. Sinkt der verfügbare phsysikalische Speicher und kommt es zu einer Auslagerung, dann steigt der Wert. Die Auslagerungsdatei sollte in etwa 1,5 bis 2 mal so groß sein wie der physikalische Speicher.
Seiten/s	Seiten/s	sar: sar -B Ausgabe	Systemstatus: Reiter Arbeitsspeicher, Button aktive Speicherpools, gewünschten Pool mit Doppelklick auswählen, Reiter Leistung, Spalte Datenbanseiten/s bzw. Nicht-Datenbanseiten/s	Windows, Linux: < 20 i5/OS: < 10 (bei "warmer" JVM), für Datenbank-Pools kann dieser Wert kurzfristig auch höher sein, da die DB2 den verfügbaren Speicher voll ausnutzt, und so viele Seiten wie möglich im Speicher behält. Deshalb ist diese Kennzahl für Datenbanken nicht sehr aussagekräftig, da Seitenein- bzw. -auslagerungen unabhängig von gerade stattfindenden Datanbankabfragen stattfinden.	Ursachen für zu hohe Pagingraten sind vor allem zuwenig physikalischer Speicher, oder auch schlechte Seitenersetzungsstrategien von Applikationen (Beispielsweise eine falsche Caching-Strategie für ein Business Object in Semiramis).
Verfügbarer physikalischer Speicher	Verfügbare MB	memory Sektion, Spalte free top: Mem Zeile sar: sar -B, Spalte kbmemfree	Systemstatus: Reiter Arbeitsspeicher, Button aktive Speicherpools, gewünschten Pool mit Doppelklick auswählen, Reiter Konfiguration, Feld Aktuell wrksyssts: Pool Size M	Windows: > 4MB Linux: > 10% i5/OS: Es müssen die momentanen Größen aller Pools auf dem System zusammengezählt werden, und dieser Wert mit der installierten Menge an physikalischem Speicher verglichen werden.	Beenden von nicht benötigten Applikationen. Auslagern von Applikationen auf einen anderen Rechner. Prüfen, ob eine Applikation (z.B. eine Semiramis Anwendung) optimiert werden kann, damit sie weniger Speicher benötigt. Prüfen, ob nahezu der vollständige Java Heap bei einem "warmen" System verwendet wird. Wenn nein, Verkleinerung des Heaps. Wenn all dies nicht möglich ist, Hinzufügen von mehr physikalischem Speicher.

I/O

Kennzahl	Windows: *perfmon*	Linux: *iostat* , df, sar	i5/OS: *iSeries Navigator,* wrkdsksts	Grenzwerte	Kommentare/ Handlungsempfehlungen
Freier Speicherplatz	Freier Speicherplatz (%)	df: df -k -al	Systemstatus: Reiter Plattenspeicherplatz, Feld Auslastung	> 15%	Installieren größerer oder weiterer Festplatten.
Auslastung	Zeit (%)	es konnten keine Linux native Tools zur Anzeige dieser Kennzahl ermittelt werden. Installation von Drittprodukten notwendig.	Systemstatus: Reiter Plattenspeicherplatz Button Plattenpools, gewünschten Pool mit Doppelklick auswählen, Spalte %ausgelastet wrkdsksts: % Used Systemüberwachung: Plattenspeicher (durchschnittlich).	Windows, Linux: < 90% i5/OS: <90%. Bei 99% schaltet sich das System ab Unterschiedliche Plattenauslastung ist sehr schlecht.	Ist die Last auf erhöhtes Paging zurückzuführen, dann Installation von mehr physikalischem Speicher. Ist die Anwendung naturgemäß I/O-lastig, dann Aufteilung der Last auf verschieden Festplatten oder Installation schnellerer Festplatten. i5/OS: Daten gleichmäßig auf alle Festplatten verteilen.
Plattenzugriffs-armauslastung	nicht verfügbar	nicht verfügbar	wrkdsksts: %Busy Systemüberwachung: Auslastung des Plattenzugriffsarms (durchschnittlich)	i5/OS: alle Zugriffsarme < 70% ein Zugriffsarm < 50%	Daten gleichmäßig auf alle Festplatten verteilen.
Durch-schnittliche Warte-schlangen-länge	Durchschnittliche Warteschlangen-länge des Datenträgers	es konnten keine Linux native Tools zur Anzeige dieser Kennzahl ermittelt werden. Installation von Drittprodukten notwendig.	nicht relevant, die Plattenzugriffsarm-auslastung ist aussgakräftiger.	Anzahl der Köpfe bzw. Spuren plus 2	Installieren schnellerer Festplatten mit mehr Köpfen.
Lese-vorgänge	Lese-vorgänge/s	Blk_read/s sar: sar- d, Spalte rd_sec/s (Achtung: Sektoren)	wrkdsksts: Read Rqs, Read (K)	Als Faustformel gelten 50-70 Lesevorgänge/s für moderne SCSI-Festplatten. Wenn die Festplatte über einen Cache Speicher verfügt, kann dieser Wert kurzfristig überschritten werden. Wird ein RAID verwendet, so ist der zusätzliche Overhead zu beachten.	Kontrolle der Spezifikationen der installierten Festplatten. Falls die durchschnittliche Datengröße von Lesevorgängen ermittelt werden kann, so kann die unter Abschnitt 2.2.1.3 beschriebene Berechnung zur Ermittlung der durchschnittlichen Zeit, um Daten von einer Festplatte abzurufen, angewendet werden. 1 dividiert durch den errechneten Wert ergibt die maximalen Lesevorgänge/s. Wenn der errechnete oder der Faustformel Wert überschritten wird, dann Installation schnellerer oder weiterer paralleler Festplatten.
Schreib-vorgänge	Schreib-vorgänge/s	Blk_wrtn/s sar: sar -d, Spalte wr_sec/s (Achtung: Sektoren)	wrkdsksts: Write Rqs, Write (K)	siehe Lesevorgänge/s	siehe Lesevorgänge/s

A.1.4 Präsentieren der Ergebnisse

<< Fassen Sie hier noch einmal die Ergebnisse der Analyse zusammen. Stellen Sie hier auch die weitere Vorgehensweise dar, falls das Problem nicht sofort behoben werden konnte. Halten Sie in diesem Fall fest, wer für den weiteren Verlauf verantwortlich ist. Verwenden Sie nach Möglichkeit Grafiken, um Ergebnisse (beispielsweise Zeitverläufe von Auslastungen oder Antwortzeiten) übersichtlich darstellen zu können>>

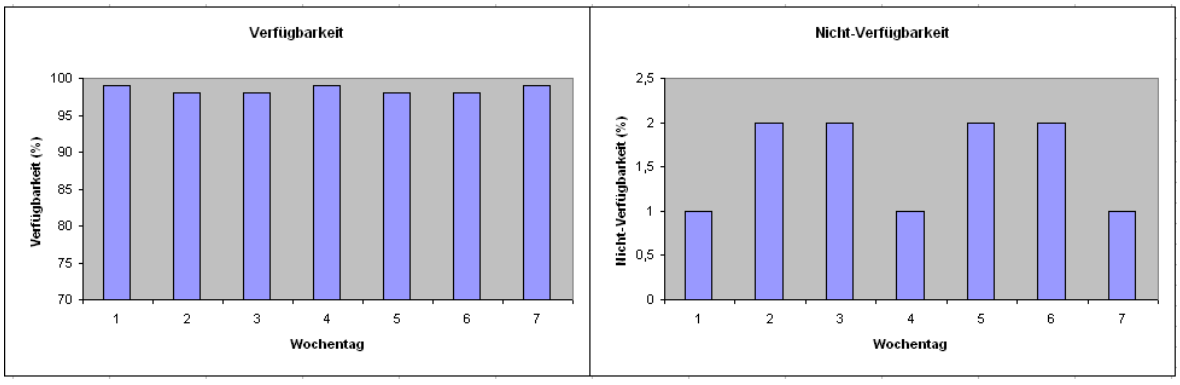
Nachfolgend sind die wichtigsten Punkte zusammengefasst, die das Verständnis von Grafiken verbessern können.

Leseaufwand minimieren: *Der Leser der Grafik sollte minimalen Aufwand zum Verständnis der Grafik aufbringen müssen. Befinden sich mehrere Kurven auf einer Grafik so ist direktes Beschriften der Kurven einer Legende zu bevorzugen.*

Information maximieren*: Beispielsweise sollen Axenbeschriftungen mit Wörtern anstatt mit Variablen erfolgen. Verwendet man Variablen, muss der Leser vorher im Dokument ihre Bedeutung nachlesen. Beschriftungen von Axen sollten so aussagekräftig wie möglich sein. Wenn man die tägliche CPU Auslastung darstellen will, soll die Axenbeschriftung auch genau so heißen.*

Tinte sparen: *Unnötige Informationen sollen weg gelassen werden, beispielsweise sollen Axenlinien weg gelassen werden, sofern sie nicht unbedingt benötigt werden.*

Geeignete Skalierung wählen: *Ein gutes Beispiel hierfür ist die Darstellung der Verfügbarkeit eines Systems an 7 Tagen der Woche. Angenommen sie liegt immer knapp unter 100%. Wenn man nun die Verfügbarkeit in einem Balkendiagramm darstellt, bekommt man 7 Balken, die alle fast bis ans Ende der Skala reichen. Stellt man aber die Nicht-Verfügbarkeit dar, und skaliert die Grafik anders, bekommt man eine viel aussagekräftigere Grafik. Siehe dazu die folgende Abbildung:*



Gängige Praktiken verwenden: *Grafiken sollten das präsentieren, was ein Leser im Allgemeinen erwartet. So soll der Ursprung der Grafik (0,0) sein, die unabhängige Variable entlang der x-*

Achse, die abhängige Variable entlang der y-Achse dargestellt werden. Skalen sollten linear (oder in besondern Fällen logarithmisch) sein, Werte sollten von links nach rechts bzw. von unten nach oben ansteigen. Ausnahmen von dieser Regel sollten nur gemacht werden, wenn es keine andere Möglichkeit gibt, da sie immer einen Mehraufwand für den Leser bedeuten.

Mehrdeutigkeit vermeiden: Axen, Skalen und Kurven sollen gekennzeichnet sein. In einem Chart sollten nach Möglichkeit nicht mehrere Variable dargestellt werden.