Q2. Messung und Benchmarking

Gliederung

1. Lasten in Computersystemen und verteilten Systemen
2. Benchmarks
3. Monitore
4. Repräsentation von Daten
5. Lastprognosen

Literatur (primär)

- R. Jain. The Art of Computer Systems Performance Analysis. Wiley 1991. Teil II (Kap. 4, 7-11)
- P. McKerrow. Performance Measurement of Computer Systems. Addison-Wesley 1987.
- David J. Lilja. Measuring Computer Performance - A Practitioner's Guide. Cambridge University Press 2000

Weiteres

- http://www.spec.org
- Folien von David Lilja http://www.arctic.umn.edu/perf-book/supplement.html

Ziele:

- Festlegung adäquater Lasttypen und –maße
- Methoden der Lastanalyse und –messung kennen lernen
- Problem der Messung in verteilten Systemen einordnen können
- Einige Benchmarks kennen lernen
- Einordnen der Möglichkeiten unterschiedlicher Monitoring-Ansätze
- Verfahren zur Zusammenfassung und Darstellung von Daten kennen lernen
- Grenzen und Chancen von Prognosemethoden erkennen

Q 2.1 Lasten in Computersystemen und verteilten Systemen

Ziele von Messungen

- Bewertung der Leistungsfähigkeit, Zuverlässigkeit eines Systems oder eines Systemteils
- Bestimmung von Parametern als Teil der Modellbildung für die Leistungs-/Zuverlässigkeitsanalyse eines Systems oder eines Systemteils
- Erlangung einer Basis für die Vorhersage über das Verhalten zukünftiger Systeme (d.h. Projektion, Prognose)

immer bezüglich der Analyseziele

⇒ Analyseziel definiert Maße, diese möglichst direkt (oder indirekt) messen, so dass Messung repräsentativ

Messung ist auf vielen Ebenen möglich!
Bsp. Rechnernetz

Ende-zu-Ende

- Ladezeit für eine Web-Seite
- RTT für ein Ping-Paket
- Übertragung einer Datei per sftp

Netzintern

- Übertragungs-zeit eines Ethernet-Frames
- Bitfehlerrate einer Verbindung

Zustand

- Auslastung eines Routers
- Verfügbar-keit einer Verbindung

➢Unterschiedliche Größen
➢Unterschiedliche Zeitskalen

} Unterschiedliche Messmethoden

Zentrale Fragen: Wie misst man? Was misst man? Wo misst man?

Wie:

- Passiv durch Beobachten
- Aktiv durch induzierte Signale/Pakete etc.
- Durch Software
- Durch Hardware
- Lokal
- Verteilt

Implikationen:

- Aufwand
- Auflösung
- Beeinflussung

Was:

Es wird eine **Metrik** benötigt!

Klassen von Maßen:

- Zähler: Wie oft tritt ein Ereignis ein
- Dauer: Wie lange dauert eine Aktion
- Größe: Welchen Umfang hat ein Parameter

Üblicherweise Skalierung bzgl. eines Intervalls:

- Transaktionen pro Zeiteinheit, Durchsatz pro Zeiteinheit

Anforderungen an eine Metrik:

- Verständlichkeit
- Messbare Basis
- Mit geringer Variabilität messbar
- Möglichst Linearität
- Ordnungserhaltend
- Konsistent über verschiedene Systeme

Wo?

Messung soll *typische Ergebnisse* für die zu untersuchende Situation liefern d.h. die Realität wiederspiegeln

- Messung im realen Betrieb (falls überhaupt möglich)
- Messung unter synthetischer Last (Benchmarks Q 2.2)

Vorteile Messung im realen Betrieb

- Reales Verhalten wird untersucht
- Keine Betriebsunterbrechung, oft nur geringfügige Beeinträchtigung

Vorteile Messung synthetischer Lasten

- Last ist kontrollierbar
- Last ist reproduzierbar
- Last ist portierbar (z.B. zum Systemvergleich)

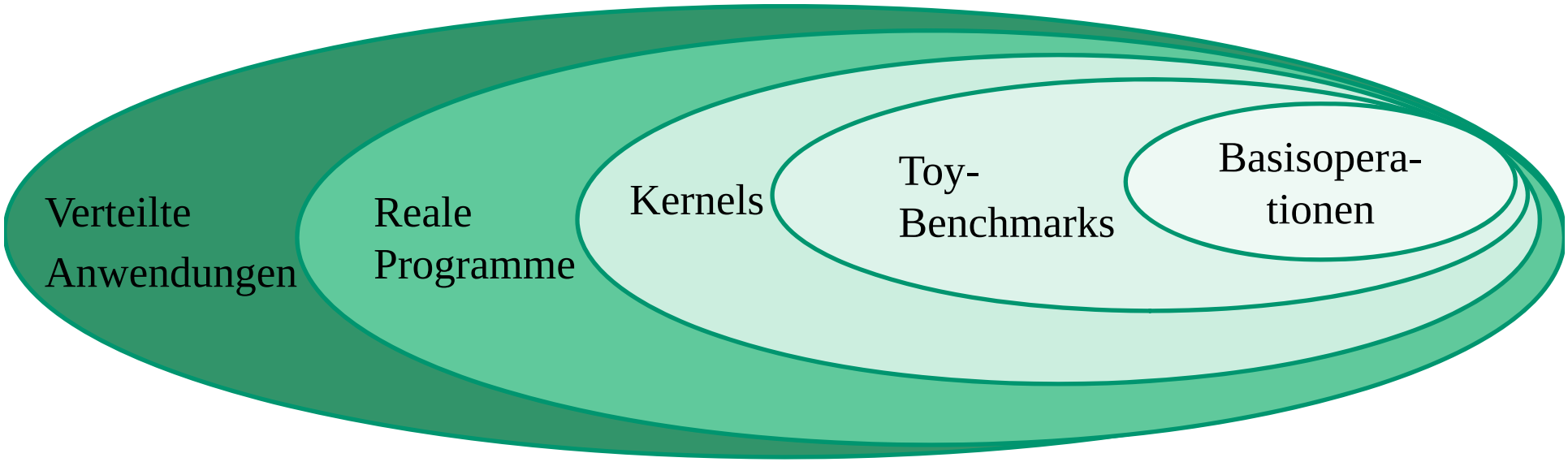
Messung am realen System

- Was soll gemessen werden?
  - I.d.R. Ankunftsrate/-intensität und Größe einer Lasteinheit (siehe SLA)
  - Alle Lasteinheiten oder Stichproben oder nur Lasteinheiten, die spezielle Bedingungen erfüllen
- Wann soll gemessen werden?
  - In Zeiträumen, in denen das System typisches Verhalten zeigt
- Wie werden die Messdaten repräsentiert ?
  - Messwerte i.d.R. statistisch, d.h. statistische Techniken verwenden (ein wenig mehr dazu gleich in Q2.4)
  - Modelltypen geben Parameter vor (z.B. nur Mittelwerte, nur Phasenverteilungen ,…)
- Wie werden Messdaten validiert?
  - Mehrfach Messen und auf statistische Ähnlichkeit testen (siehe auch Vorlesung MAO)

Q 2.2 Benchmarks

- Benchmarks sind synthetische Programme speziell zur Messung von Systemen (Computersystemen, -netzen etc.)

Benchmarkhierarchie

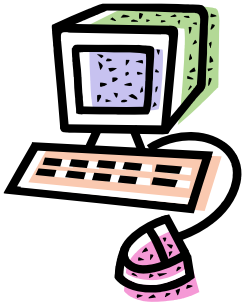


- Benchmarks für Basisoperationen
  Untersuchung der Geschwindigkeit einzelner Operationen wie Addition, Multiplikation, .. (z.B. Dhrystone, Whetstone, …)
- Toy-Benchmarks
  kleine klassische Programme mit geringer praktischer Relevanz
  (z.B. Türme von Hanoi, ….)
- Kernel-Benchmarks
  Teile realer Programme, die besonders viel Zeit verbrauchen
  (z.B. Livermore Loops, Linpack, …)
- Reale Programme
  Komplexe Programme aus vorgegebenen Anwendungsgebieten
  (z.B. TPC-C, SPEC-cpu, SPEC-viewperf,…)
- Verteilte Programme
  Verteilte Anwednungen, oft Web-Server oder verteilte DB
  (z.B. TPC-App, SPEC-web, SPECjAppServer, …)

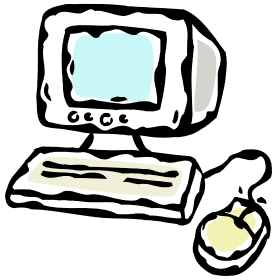
Wesentlicher Einsatzbereich von Benchmarks: Systemvergleich

Sei t(A,P,S) Zeit zur Abarbeitung von Benchmark A mit Parametern P im System S

S



S‘



t(A,P,S) < t(A,P,S‘)

S ist besser als S‘?

Offene Fragen:

- Sind die Unterschiede signifikant?
- Gilt evtl. t(A,P‘,S) > t(A,P‘,S‘) oder t(B,P,S) > t(B,P,S‘)
  Welcher Benchmark/ Parametersatz spiegelt am besten die reale Last wieder?
- Misst de Benchmark das richtige Leitungsmaß?
  Was ist mit den Kosten und dem Energieverbrauch?

Einfache Benchmarks (Basisoperationen bis Kernels):

- Sind verständlich und nachvollziehbar
- Weisen keine/wenige Parameter auf
- Sind einfach zu installieren und zu messen
- Sind aber oft nicht repräsentativ

Komplexe Benchmarks (reale und verteilte Programme)

- Sind aufwändig zu parametrisieren und zu installieren
- Sind kaum nachvollziehbar
- Erlauben eine Vielzahl unterschiedlicher Messungen
- Können so konfiguriert werden, dass sie repräsentativ sind

Gefahr beim Vergleich mittels etablierter Benchmarks:
Lokale Optimierung durch Hersteller oder Betreiber!

Beispiel TOP 500 Liste

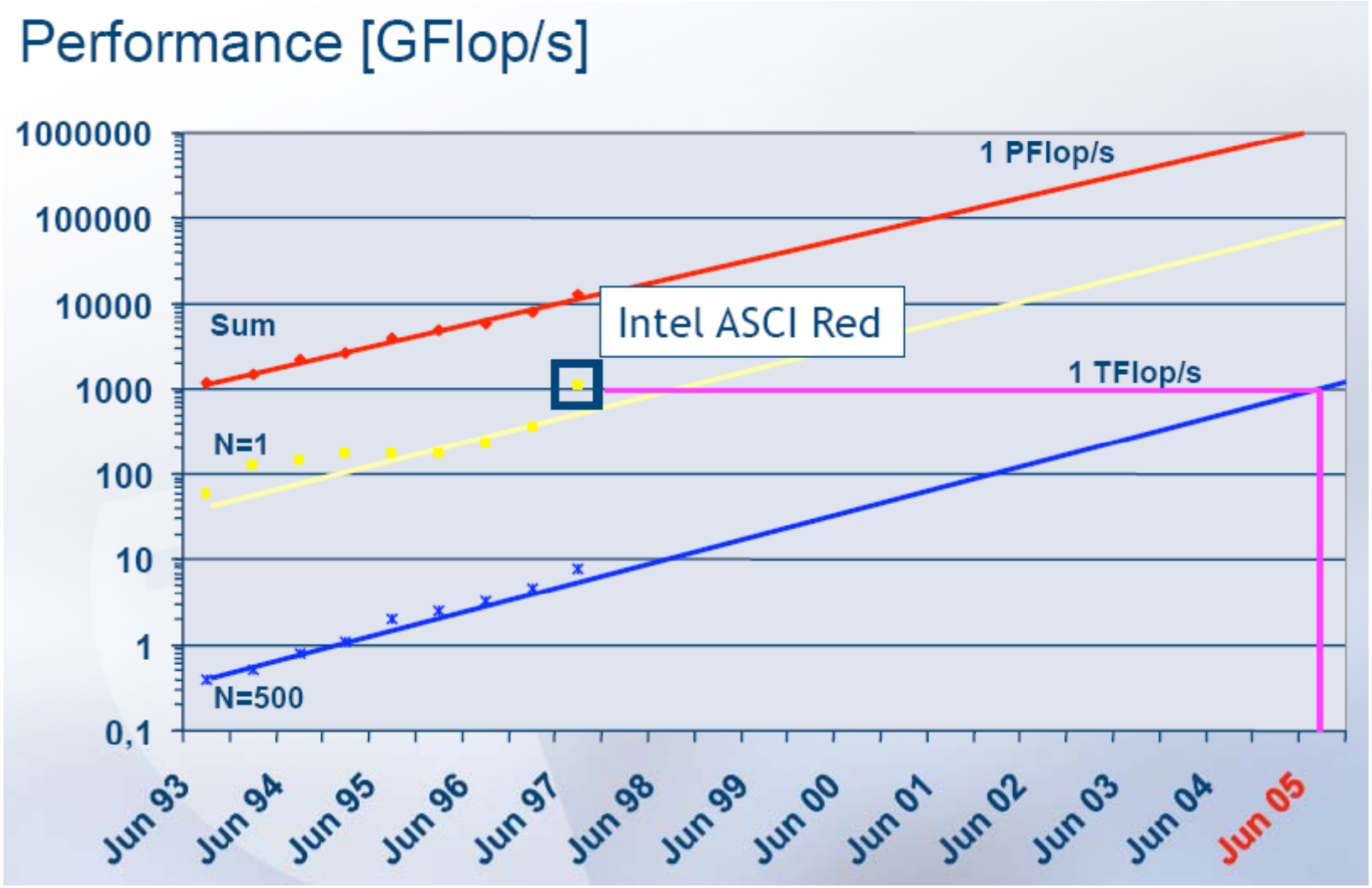
- Liste der 500 schnellsten Supercomputer, seit 1993 halbjährlich veröffentlicht
- Basis: Lösung des Gleichungssystems Ax = b mit einer dichten Matrix mittels der Gauß-Elimination
  (LINPACK Programm, relativ altes FORTRAN-Programm)
- Leistungsmaß: R_{max} Linpack Durchsatz im Gegensatz zu R_{peak} theoretischer Maximaldurchsatz auf Grund der Prozessorleistungen
  weiterhin relevant N_{max} Größe des Gleichungssystems für das R_{max} erreicht wurde und $N_{1/2}$ Größe des Gleichungssystems für die R_{max} /2 erreicht wird
- Klare Vorgaben, was im Algorithmus geändert werden darf (zur Parallelisierung)

Einfaches Vorgehen, oft kritisiert aber mit großer Wirkung!

Die ersten 10 der aktuellen Liste (Juni 09):

Computer	Country	Year	Cores	RMax	RPeak	Nmax	Nhalf	Power
BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband	United States	2008	129600	1105000	1456700	2329599	0	2483,47
Cray XT5 QC 2.3 GHz	United States	2008	150152	1059000	1381400	4712799	0	6950,6
Blue Gene/P Solution	Germany	2009	294912	825500	1002700	0	0	2268
SGI Altix ICE 8200EX, Xeon QC 3.0/2.66 GHz	United States	2008	51200	487005	608829	2300760	0	2090
eServer Blue Gene Solution	United States	2007	212992	478200	596378	2456063	0	2329,6
Cray XT5 QC 2.3 GHz	United States	2008	66000	463300	607200	0	0	0
Blue Gene/P Solution	United States	2007	163840	458611	557056	0	0	1260
SunBlade x6420, Opteron QC 2.3 Ghz, Infiniband	United States	2008	62976	433200	579379	0	0	2000
Blue Gene/P Solution	United States	2009	147456	415700	501350	2958335	0	1134
Sun Constellation, NovaScale R422-E2, Intel Xeon X5570, 2.93 GHz, Sun M9/Mellanox QDR Infiniband/Partec Parastation	Germany	2009	26304	274800	308283	0	0	1549

Entwicklung:



Methodik zeigt die Entwicklung und bildet eine Rangfolge (einfach, verständlich, publizierbar, ..)

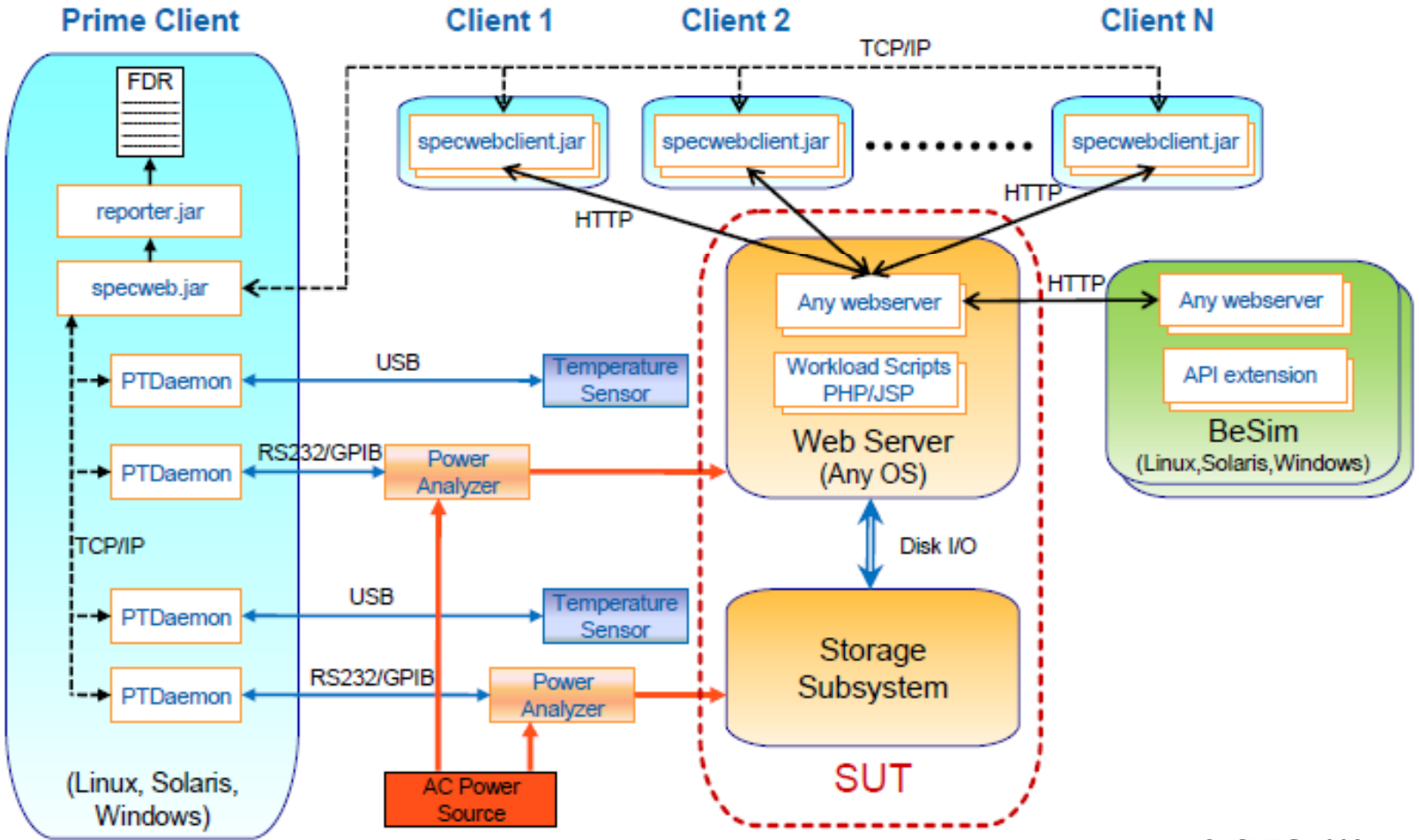
Aber erlaubt keine Aussagen über Systemleistung und erst recht keine Systemanalyse

$\Rightarrow$Nicht für die Ziele der Vorlesung geeignet

Hier verwendet

- Einfache Benchmarks auf Komponentenebene
  - Messung der Übertragungsrate durch Paketströme
  - Messung der Zugriffszeit von Platten durch Datenströme (Mix aus Lese- und Schreibanforderungen)
- Komplexe Benchmarks auf Systemebene
  - Für spezielle Anwendungen entwickelt
  - Falls kein Benchmark vorhanden, u.U. eigene Anwendungen verwenden

Beispiel SPECweb (seit 99 aktuelle Version 2009)



© SPEC 2009

Drei Anwendungsszenarien:

- Bankanwendung
  Sehr dynamisch, vollständig über HTTPS
- E-Commerce Anwendung
  Sehr dynamisch, teilweise HTTP, teilweise HTTPS
- Standard-Anwendung
  Relativ statisch, laden von Web-Seiten, ohne SSL

Eigenschaften:

- Caching wird nachgebildet
- PHP und JSP Implementierungen sind enthalten
- Bilder werden über 2 parallele HTTP-Verbindungen geladen

Benutzer kann den Benchmark nicht modifizieren (keine Parameter!)
Konfiguration durch Anzahl parallel laufender Clients

Wie viele Clients können parallel auf den Web-Server zugreifen, so dass eine vorgegebene Dienstqualität erreicht wird ? (= SPECwebNumber)

Dienstqualitäten:

- Bank und E-Commerce: 95% der Antworten in weniger als 2 Sekunden, 99% in weniger als 4 Sekunden
- Normal: Benutzer versucht Seite mit 100.000 Bytes/sek. zu laden, 95% der Seiten müssen mit 99.000 Bytes/Sek. geladen werden, 99% mit 95.000 Bytes/Sek.

Angabe der Leistungsmaße in Relation zum Referenzsystem

$$SPECwebNumber = \sqrt[3]{\frac{EComm\_Numb}{EComm\_Ref} \cdot \frac{Bank\_Numb}{Bank\_Ref} \cdot \frac{Normal\_Numb}{Normal\_Ref}}$$

Vergleichsresultate auf Basis der SPECwebNumber:

SPECweb2009_JSP (5):

Test Sponsor Hardware Vendor	System	Web Server Software	Processor					Total Memory (GB)	Submeasurements		
			CPU Description	MHz	Chips	Cores	Total Threads		Banking sus @ W	Ecommerce sus @ W	Support su @ W
Fujitsu Fujitsu	PRIMERGY TX150 S6 HTML	Rock Web Server v1.4.7 (x86_64), Rock JSP/Servlet Container v1.3.2 (x86_64)	Intel Xeon L3360	2833	1	4	4	8	27300 sus 188 W	32300 sus 185 W	14100 su 176
Hewlett-Packard Hewlett-Packard	HP ProLiant DL370 G6 HTML	Rock Web Server v1.4.7 (x86_64), Rock JSP/Servlet Container v1.3.2 (x86_64)	Intel Xeon W5580	3200	2	8	16	96	100032 sus 736 W	124200 sus 716 W	70400 su 722
Hewlett-Packard Hewlett-Packard	HP ProLiant DL370 G6 HTML	Rock Web Server v1.4.7 (x86_64), Rock JSP/Servlet Container v1.3.2 (x86_64)	Intel Xeon W5580	3200	2	8	16	48	103104 sus 691 W	133440 sus 655 W	58048 su 633
Hewlett-Packard Hewlett-Packard	HP ProLiant DL370 G6 HTML	Rock Web Server v1.4.7 (x86_64), Rock JSP/Servlet Container v1.3.2 (x86_64)	Intel Xeon W5580	3200	2	8	16	48	108096 sus 419 W	139200 sus 430 W	66560 su 379
Hewlett-Packard Hewlett-Packard	HP ProLiant DL380 G6 HTML	Rock Web Server v1.4.7 (x86_64), Rock JSP/Servlet Container v1.3.2 (x86_64)	Intel Xeon L5520	2266	2	8	16	48	78336 sus 351 W	109440 sus 343 W	51520 su 326

Quelle: www.spec.org

Detaillierte Resultate:

Banking									
Simultaneous User Sessions	Test ITeration	Minimum QOS Compliance			Validation Errors	Average Active Power (W)			SUS/Watt
		Good	Tolerable	Fail		Server	Storage	Sum	
100032	1	99.2	100	0.00	0	464	272	736	136
	2	98.3	100	0.00	0	463	273	736	136
	3	98.6	100	0.00	0	465	273	737	136
Average						464	272	736	136
Alterable / Modifiable Configuration	Used Components					Unused Components			
	25x72GB SFF 15k SAS(HW RAID0,ext2)					None			

Ecommerce									
Simultaneous User Sessions	Test ITeration	Minimum QOS Compliance			Validation Errors	Average Active Power (W)			SUS/Watt
		Good	Tolerable	Fail		Server	Storage	Sum	
124200	1	99.6	100.0	0.00	0	471	243	714	174
	2	99.3	100.0	0.00	0	473	244	717	173
	3	98.7	99.7	0.33	0	473	244	717	173
Average						472	244	716	173
Alterable / Modifiable Configuration	Used Components					Unused Components			
	6x72GB SFF 15k SAS(HW RAID0,ext2)					19x 72GB SFF SAS (HW RAID0, ext2, mounted on /mnt/spare)			

Quelle: www.spec.org

Konfigurationsbeschreibung (Ausschnitt):

Availability Dates	
SUT Hardware	Jun-2009
Backend Simulator	Jul-2005
Web Server Software	May-2008 May-2008
Operating System	Jan-2009
Other Components	Apr-2009 (JVM)

System Under Test (SUT)	
# of SUTs	1
Vendor	Hewlett-Packard
Model	HP ProLiant DL370 G6
Processor	Intel Xeon W5580
Processor Speed (MHz)	3200
# Processors	8 cores, 2 chips, 4 cores/chip, 16 threads
Primary Cache	32 KB I + 32 KB D on chip per core
Secondary Cache	256 KB I+D on chip per chip
Other Cache	8 MB I+D on chip per chip
Memory	96 GB (12x8GB PC3-10600R DDR3)
Disk Subsystem	2x36GB SFF SAS 15K RPM, 27x72GB SFF SAS 15K RPM
Disk Controllers	Smart Array P410i (embedded), Smart Array P411 Controller with 512MB cache
Operating System	RedHat Enterprise Linux 5.3 (2.6.18-128.el5)
File System	ext2
Other Hardware	1x Modular Smart Array 70 Enclosure, 2xHP ProCurve Switch 3400cl-48G (J4906A), 2xHP Procurve Media Flex Module (J8435A), 4x HP Procurve SR Transceiver (J8436A)
Other Software	Java(TM) SE Runtime Environment (build 1.6.0_13-b03), Java HotSpot(TM) 64-Bit Server VM (build 11.3-b02, mixed mode)

Web Server Software	
Vendor	Accoria Networks, Inc.
Name/Version	Rock Web Server v1.4.7 (x86_64)
Dynamic Scripts	JSP
Server Cache	N/A
Log Mode	Binary Common Log Format

JSP Script Engine	
Vendor	Accoria Networks, Inc.
Name/Version	Rock JSP/Servlet Container v1.3.2 (x86_64)
Dynamic Scripts	JSP
Server Cache	N/A
Log Mode	Binary Common Log Format

Clients	
# of Clients	64
Model	HP ProLiant DL360 G3
Processor	Intel Xeon
Processor Speed (MHz)	3060
# Processors	2 cores, 2 chips, 1 core/chip, 4 threads (Hyper-Threading Enabled)
Memory	4 GB
Network Controller	NC7170 (embedded)
Operating System	Windows Server 2003 SP1
JVM Version	Java 2 Runtime Environment, Standard Edition (build 1.5.0_06-b06)
JIT Version	Java HotSpot Client VM (build 1.5.0_06-b05, mixed mode, sharing)
Other Hardware	HP ProCurve 2848 for client management on second embedded NIC
Other Software	N/A

Zusammenfassung SPEC (und ähnliche Benchmarks):

- Installation, Ausführung und Messung sind aufwändig
- Ergebnisse im jeweiligen Anwendungskontext meistens repräsentativ
- Messergebnisse auch zur Modellparametrisierung nutzbar (dann aber oft detailliertere Messung und Modellierung des Benchmarks notwendig (siehe Q4))
- Es existieren viele weitere SPEC-Benchmarks:
  - SPEC-CPU
  - SPECviewerperf
  - SPEC MPI
  - SPCjms
  - u.v.a

Q 2.3 Monitore

Ein Monitor ist ein Tool zur Beobachtung von Aktivitäten in einem System

Idealerweise sollte ein Beobachter das Systemverhalten beobachten aber in keiner Weise beeinflussen

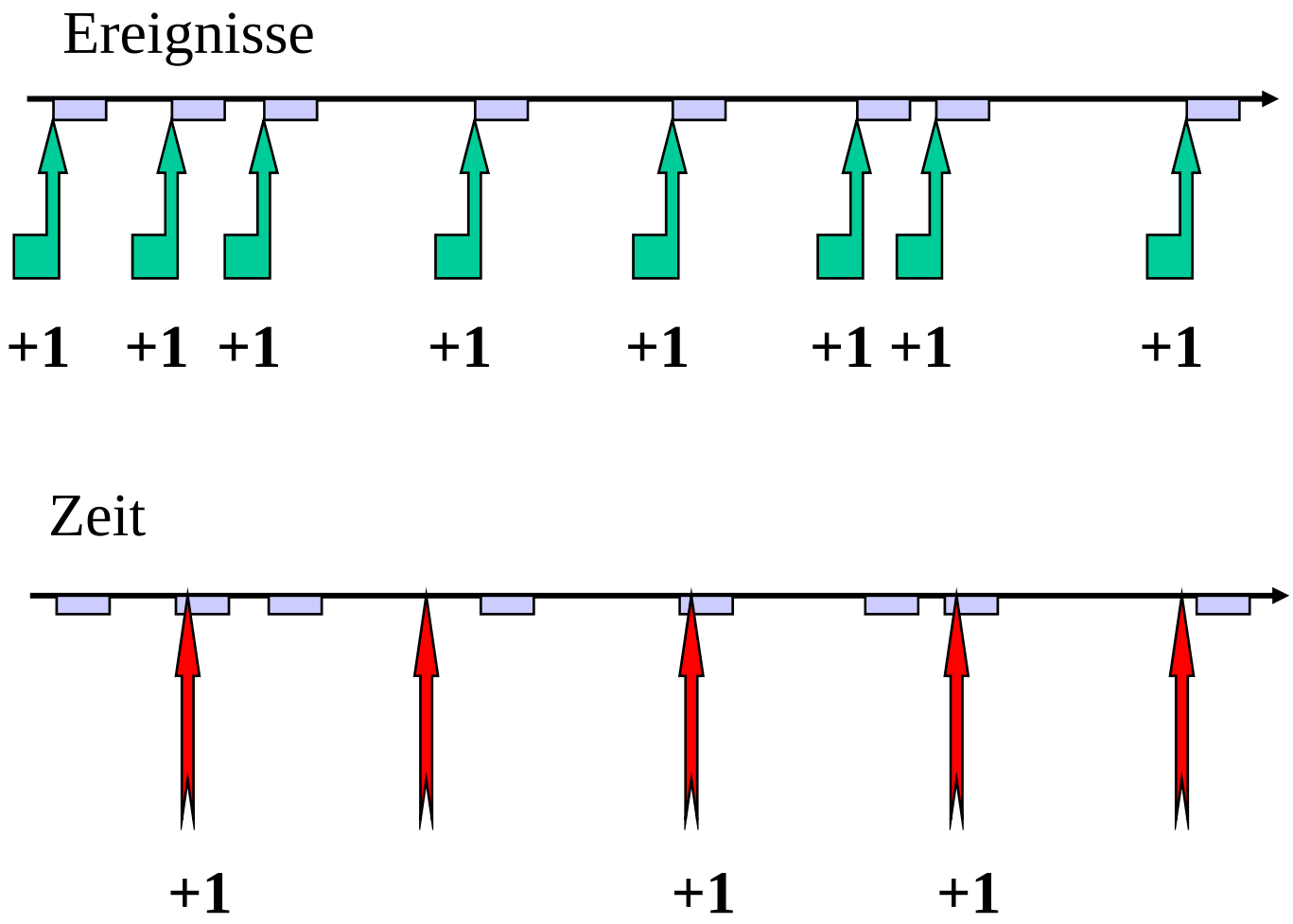
Man unterscheidet:

- Online Monitore: Messung und Auswertung zur Laufzeit
- Offline Monitore: Datensammlung zur Laufzeit, Auswertung später

Monitortypen:

- Software
- Hardware
- Hybrid

Messung kann ereignis- oder zeitgesteuert erfolgen



Beispiele für Ereignisse:

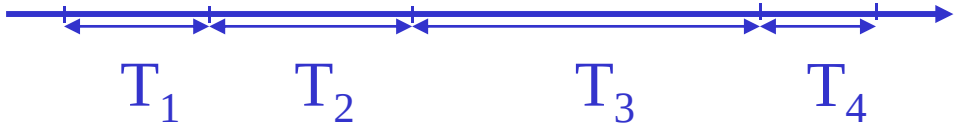
- Plattenzugriff
- Prozessor-Interrupt
- Paketübertragung
- ..

Zeit- und ereignisgesteuerte Messungen nutzen Timer

Timer werden als Hardware- oder Software-Zähler realisiert

- ➢ Takt gibt die Messgenauigkeit vor
- ➢ Zugriff erfordert Unterprogrammaufruf + Lesen einer Speicherzelle

- T_1 = Lesezeit für den Wert des Counters
- T_2 = Speicherzeit des Wertes
- T_3 = Ausführungszeit Ereignis
- T_4 = Lesezeit für den Wert des Counters
  - $T_4 = T_1$



➢ Overhead $T_{ovh} = T_1 + T_2$ falls $T_{ovh} \gg T_1 + T_2$ kein Problem, andernfalls ungenaue Messungen

T_1 und T_2 schwanken in der Regel

Softwaremonitore

- Software auf Benutzer- oder Systemebene
- Zugriff auf alle Informationen, die Nutzerprogrammen oder dem Betriebssystem zur Verfügung stehen
- Aktivierung:
  - Zeitgesteuertes Auslesen von Systeminformationen alle x Sekunden
  - Ereignisgesteuerte Aktivierung bei Ausführung bestimmter Operationen (z.B. Ein-/Ausgabe)
    ($\Rightarrow$ vom Betriebssystem zu aktivieren)
- Daten können direkt ausgewertet und aggregiert werden
- Aber als Programm mit anderen Programmen in Konkurrenz um Ressourcen

Hardwaremonitore

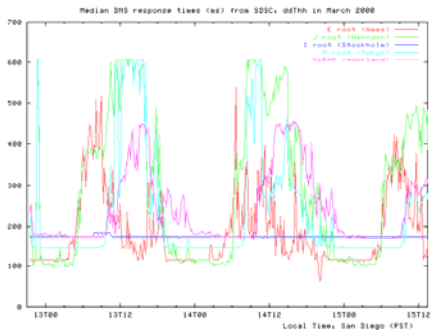
- Eigenständige Hardware zur Messung elektrischer Signale
- Messung festgelegter Signale der Hardware
  (z.B. Pakete über ein Ethernet, Signale auf dem Systembus)
  dazu müssen Messpunkte existieren
- Aktivierung:
  - i.d.R. Ereignisgesteuerte Aktivierung
- Kein Bezug zur Anwendung
- Als eigenständige Hardware kein zusätzlicher Ressourcenverbrauch auch bei hoher Eingangsrate und Auflösung

Hybride Monitore

- Kombination aus Hardware- und Softwaremonitoren

Netzwerkmonitore

- Grundsätzlich auch wieder Hardware oder Software
- Passive Messungen durch Auslesen von Paketinformationen
  - Netramet: Messung auf Routern (z.B. DNS-Messung)



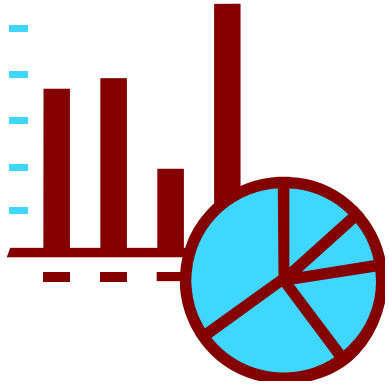
  - Tcpdump, tcptrace: Ausgabe und Auswertung von TCP-Headern am Interface
- Aktive Messungen durch Injektion von Nachrichten
  - Ping: Messung von RTT
  - Timeit: Zugriffszeiten auf Web-Seiten

Log-Dateien

- Viele Programme schreiben Log-Dateien oder können so konfiguriert werden, dass sie Log-Dateien schreiben (z.B. Web-Server, DB-Systeme)
  - Einfache Methode, um an Daten zu gelangen
  - Aber eingeschränkter Messbereich (nur aus dem Benutzerprogramm heraus)
  - Zusätzlicher Aufwand durch Dateizugriffe
- Interpretation manuell oder durch spezielle Bearbeitungs- und Auswertungsprogramme

Q 2.4 Repräsentation von Daten

- Zentrale Frage:
  Wie sollen verfügbare Daten dargestellt werden?
- Anforderungen:
  - Repräsentativ
  - Informativ
  - Relevant
  - Kompakt
  - In Modellen/Analysen nutzbar



Darstellung gemessener Werte

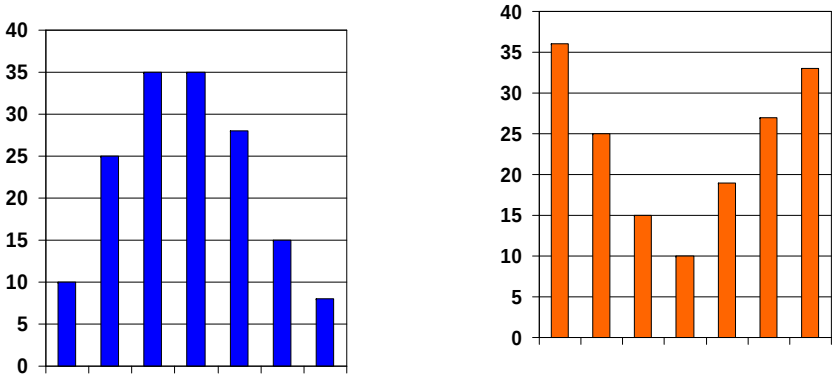
Messungen liefert meistens sehr viele Werte

- hoher Speicherbedarf
- kaum interpretierbar
- Messwerte aggregieren

Durch Maßzahlen:

- Mittelwert (arithm., geomet.)
- Median, Quantile
- Momente
- ……

Durch Histogramme:



Darstellungsalternativen in Modellen:

- Als Konstante
  (z.B. durch den Mittelwert repräsentiert)
- Als theoretische Verteilung
  - Diskrete Verteilung
  - Kontinuierliche Verteilung

  in beiden Fällen Verteilungsbestimmung, Parameterschätzung, Anpassungstest
- Als empirische Verteilung
  - Diskrete empirische Verteilung
  - Kontinuierliche empirische Verteilung (per Interpolation)
- Als stochastischer Prozess

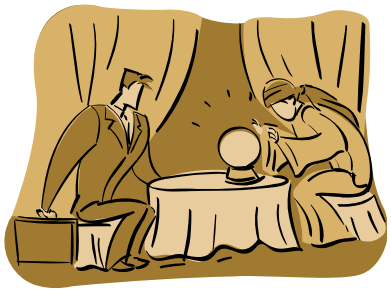
Siehe auch MAO

- Durch Maximal-/Minimalwerte
- Durch Intervalle
- Als vollständiger Trace (u.U. nach Löschen von Ausreißern)

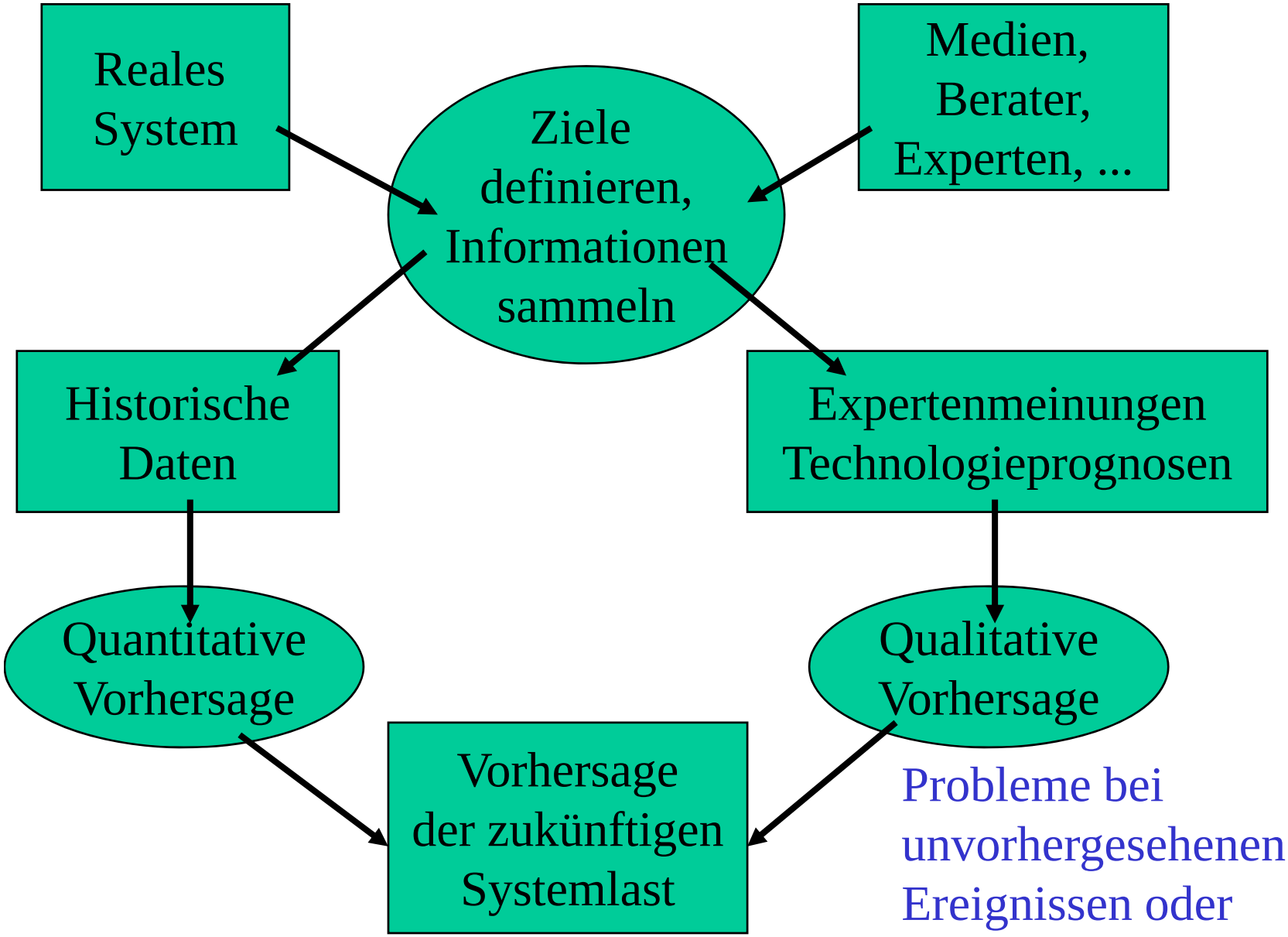
Entscheidung für eine Darstellungsform:

- Ziel der Analyse
  (z.B. Schranken, Intervalle bei Realzeitsystemen, stochastische Beschreibung zur Leistungsanalyse)
- Qualität und Umfang der Daten
  (z.B. geringer Datenumfang erlaubt keine Anpassung stochastischer Prozesse)

Q 2.4 Lastprognosen



- Modellierung und Analyse für zukünftige Systeme, Konfigurationen etc.
  Wie bestimmt man Parameter für interessante, fiktive Szenarien oder zukünftige Entwicklungen ?
- Man unterscheidet
  - Qualitative Methoden
    subjektive Formulierung von Erwartungen über die Zukunft auf Basis von
    Intuition, Expertenmeinungen, historischen Analogien, technischen Entwicklungen
  - Quantitative Methoden
    objektive mathematische Verfahren auf Basis historischer Daten und Bildung eines Modells



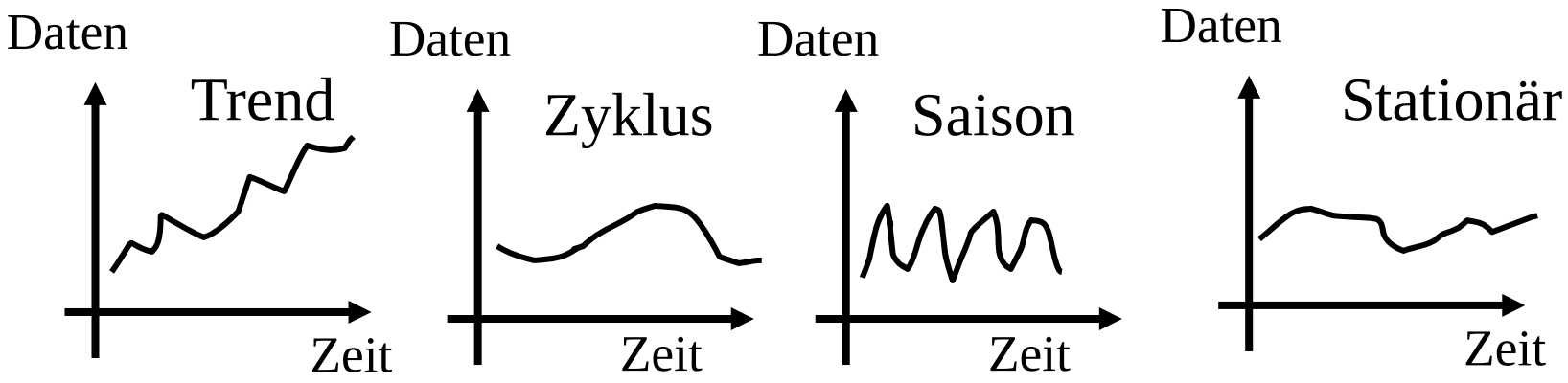
Probleme bei unvorhergesehenen Ereignissen oder Technologiesprüngen

Patterns, Entwicklungsmuster

- Muster haben starken Einfluss auf die Auswahl und Güte von Prognoseverfahren.

  Es werden (im Wesentlichen) 4 Muster unterschieden:

  – Trend: Werte wachsen (oder sinken) über die Gesamtentwicklung hinweg
  – Zyklus: Schwankungen wiederholen sich
  – Saisonale Schwankungen: wie beim Zyklus, nur ist die Wiederholungsperiode ist kürzer
  – Stationärität: kein Trend, Mittelwert über die Datenreihe hinweg ist konstant



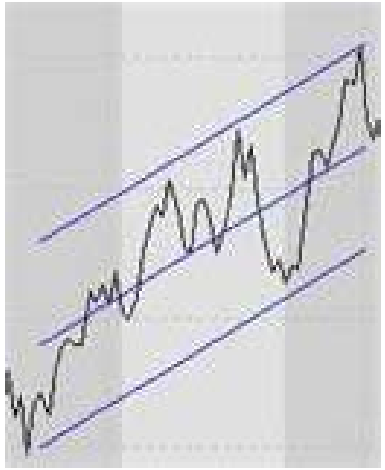
Regressionsmethoden

- Grundidee: Regressionsmodelle schätzen den Wert einer (abhängigen) Variable als Funktion einer Menge von (unabhängigen) Variablen.
- Die funktionale Form der Abhängigkeit wird vorgegeben

Entscheidungsmöglichkeiten

- Welche Variablen haben Einfluss auf das Resultat, auf die abhängige Variable, die ich prognostizieren möchte?
- Wie viele unabhängige Variablen sollen einfließen ?
  Einfachster Fall: 1 unabhängige Variable
- Von welchem Typ soll der funktionale Zusammenhang sein ?
  Einfachster Fall: linear

- Lineares Modell
  - unabhängige Variable x, abhängige Variable y
  - Regressionsgerade y = a + bx
  - Bestimmung von a und b mit der Methode der kleinsten (Fehler-)Quadrate



$$b = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}$$

$$a = \bar{y} - b\bar{x}$$

$x_i,\ y_i$ sind die gemessenene Daten

$\bar{x},\ \bar{y}$ sind die Mittelwerte, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,\ \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

Weitere einfache Ansätze:

Gleitender Durchschnitt

- Die Entwicklung einer Variable wird lediglich über die Zeit aus sich selbst heraus erklärt.
- Schätzwert für den nächsten zukünftigen Wert ist der Mittelwert der letzten n Daten:

$$f_{t+1} = \left(y_t + y_{t-1} + \ldots + y_{t-n+1}\right) / n$$

wobei f_{t+1} prognostizierter Wert für Zeitpunkt t+1
y_t beobachteter Wert zum Zeitpunkt t
n Anzahl Beobachtungen, die Einfluss haben sollen

Exponentielle Glättung

- Grundidee: ähnlich wie bei gleitenden Durchschnitten nur werden Vergangenheitswerte unterschiedlich gewichtet
- Hypothese: neuere Werte sind von größerer Relevanz als ältere Werte

$$f_{t+1} = f_t + \alpha\left(y_t - f_t\right)$$

wobei f_{t+1} prognostizierter Wert für Zeitpunkt t+1
y_t beobachteter Wert zum Zeitpunkt t
α Glättungsfaktor