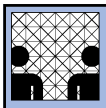
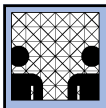


Transaktionale Informationssysteme

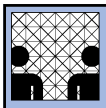
9. Transaktionssysteme

Norbert Ritter
Datenbanken und Informationssysteme
vsis-www.informatik.uni-hamburg.de



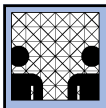
TAS: Transaktionssysteme (1)

- Unser Ziel: Ableitung von Schichtenmodellen für TA-Systeme
  - Schrittweise Abstraktion von einem Bitstring auf der Magnetplatte zu der Abwicklung von Transaktionen, die durch TACs (TA-Codes) aufgerufen werden
  - Prinzipieller Aufbau durch Schichtenbildung: Als konkrete Realisierungen ergeben sich Client/Server-Systeme
- Bereiche typischer TA-Anwendungen
  - Kommunikationssysteme
    - Telefonanrufe sind TA
      - Verbindungsaufbau und -freigabe (inkl. Betriebsmittel)
      - Anrufweiterleitung, Benachrichtigungen (Voice Mail), ...
      - komplexe Suche bei 800/900-Nummern
      - Bezahlung kann mehrere Telefon-Firmen betreffen
    - Web: e-Commerce, e-Business, ...



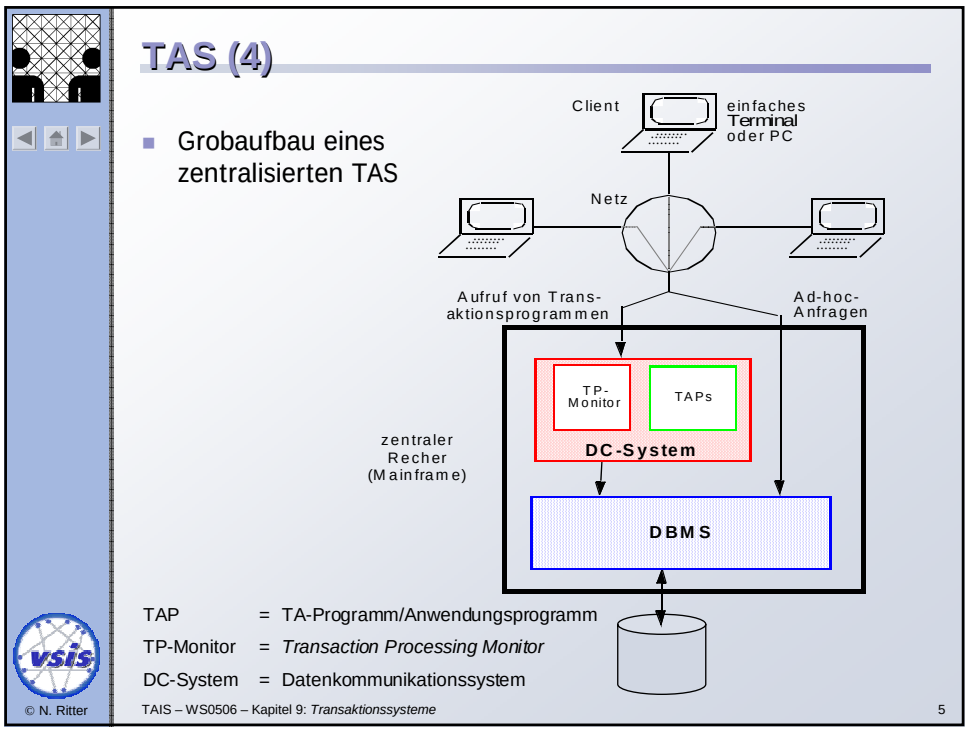
TAS (2)

- Bereiche typischer TA-Anwendungen (Forts.)
  - Finanzwelt / Banken
    - Point-of-Service-Terminals
      - Kreditkartenvalidierung
      - Direktbuchung
    - Aktienhandel
  - Reisebüro
    - Buchungen
    - Fahrkartenverkauf
  - Produktion
    - Bestellverarbeitung, Einsatz- und Lagerplanung, ...
    - CIM integriert verschiedenartige TA-Anwendungen
      - Just-in-time-Lagerkontrolle
      - Werkzeug- und Werkstück-Transport
      - automatische Handhabungssysteme
      - Qualitätskontrolle, ...



TAS (3)

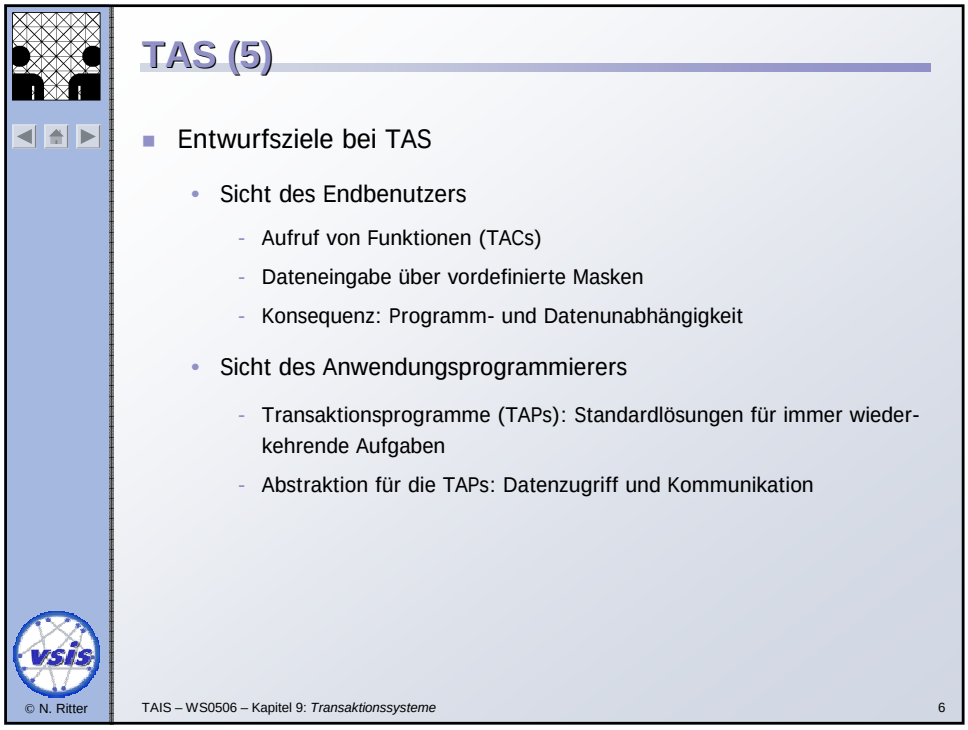
- Bereiche typischer TA-Anwendungen (Forts.)
  - Prozesskontrolle
    - Chemische Abläufe / Reaktionen
    - Energieerzeugung und -verteilung
    - automatisierte Warenhäuser
    - Flugzeuge, ...
  - Trend zum Einsatz von Standard-SW und TA-Systemen



TAS (4)

- Grobaufbau eines zentralisierten TAS

TAP = TA-Programm/Anwendungsprogramm
TP-Monitor = *Transaction Processing Monitor*
DC-System = Datenkommunikationssystem



TAS (5)

- Entwurfsziele bei TAS
  - Sicht des Endbenutzers
    - Aufruf von Funktionen (TACs)
    - Dateneingabe über vordefinierte Masken
    - Konsequenz: Programm- und Datenunabhängigkeit
  - Sicht des Anwendungsprogrammierers
    - Transaktionsprogramme (TAPs): Standardlösungen für immer wiederkehrende Aufgaben
    - Abstraktion für die TAPs: Datenzugriff und Kommunikation

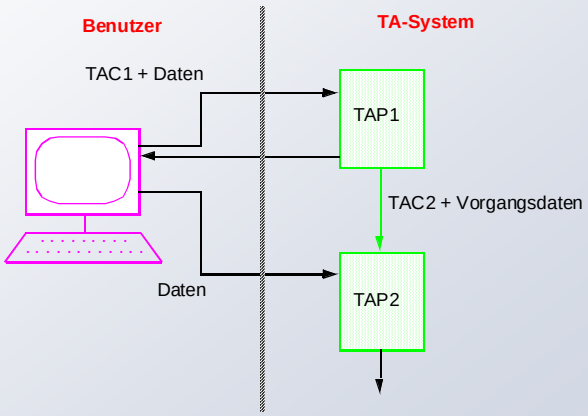
TAS (6)

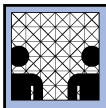
- Entwurfsziele bei TAS (Forts.)

DB-System	DC-System
- logische Sicht auf die Daten	- logische Sicht auf die Terminals
- Isolation der Programme	- Isolation der Programme
• von den Zugriffspfaden und Speicherungsstrukturen *(Datenunabhängigkeit)*	• von den Kommunikationspfaden und Terminaleigenschaften *(Kommunikationsunabhängigkeit)*
• von den Maßnahmen zur Sicherung und zum Schutz vor Wechsel-wirkungen *(Kontrollunabhängigkeit)*	• von den Techniken des Multi-Threading innerhalb eines Prozesses *(Kontrollunabhängigkeit)*

TAS (7)

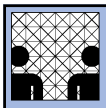
- Operationale Eigenschaften
  - Prinzipieller Aufbau





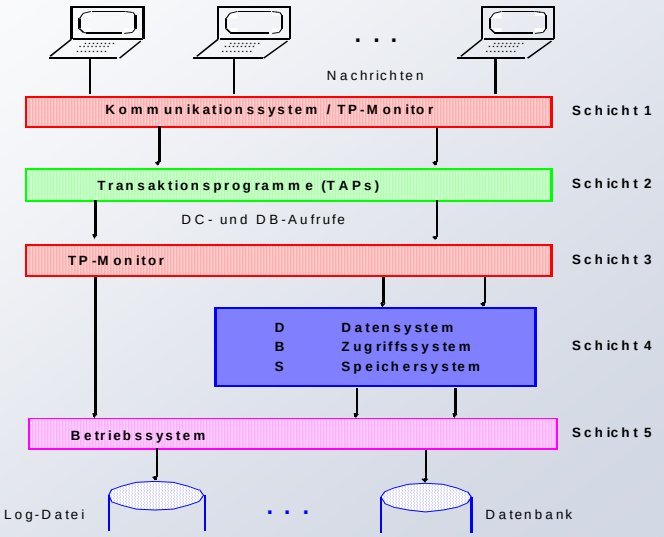
TAS (8)

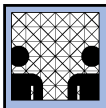
- Operationale Eigenschaften (Forts.)
  - System- und Betriebsmerkmale
    - Benutzung im Dialog: „parametrischer“ Benutzer
    - wenige kurze Transaktionstypen: hohe Wiederholrate
    - sehr viele Benutzer gleichzeitig
    - gemeinsame Datenbestände mit größtmöglicher Aktualität
    - kurze Antwortzeiten als Optimierungsziel vor Auslastung
    - stochastisches Verkehrsaufkommen
    - hohe Verfügbarkeit des Systems



TAS (9)

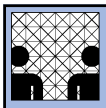
- Schichtenmodell





TAS (10)

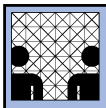
- Größe von TAS
  - Systemadministrator muss Tausende von Benutzern sowie alle HW- und SW-Komponenten verwalten; er ist für die Verfahrens-entscheidungen im Betrieb und für die Systemplanung zuständig
  - Operator betreibt das System (Aufzeichnung des Systemverhaltens, Fehlerbehandlung bei Geräten, Datenarchivierung, Installation neuer SW)
  - Wachsende Tendenz, Operatoraufgaben zu automatisieren, um Fehler zu reduzieren und Kosten zu sparen
  - Repräsentative Größenangaben
    - Siehe Tabelle auf folgender Folie
    - Ausfallzeiten?
    - Komplexe Aufgaben bei 10^5 — 10^6 Komponenten
      - Fehlersuche und -diagnose
      - Planen von Änderungen, Systemevolution
      - Legacy-Problem



TAS (11)

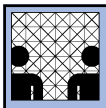
- Größe von TAS (Forts.)
  - Repräsentative Größenangaben für kleine und große TAS

Hardware (nach Gray/Reuter)	**klein**	**mittel**	**komplex**
Terminals oder Benutzer	100	10,000	100,000
Rechner	1	10	100
Platten (Kapazität)	10 (= 1 TB)	100 (= 10 TB)	1K (= 100 TB)
Archivbänder (Kap.)	1K (= 10 TB)	10K (= 100 TB)	100K (= 1 PB)
Software			
TAPs, Reports, Masken, DB-Dateien	400	4,000	40,000
Quellen und alte Versionen von Programmen, Reports, Masken	1,000	10,000	100,000
Wertebereiche/Attribute	1,000	10,000	100,000



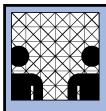
TAS (12)

- Anforderungen
  - Abwicklung sehr hoher TA-Raten
    - sehr große TA-Systeme: z.B. 150.000 nebenläufige Benutzer/System auf CICS
    - TA-Typ „Kontenbuchung“: Maßeinheit bei den Benchmarks TPC-A, -B: n Ktps vom Typ TPC-A/B?
    - Komplexere Transaktionen „Abwicklung von Bestellungen“ beim TPC-C: $n*10^2$ Ktpm vom Typ TPC-C
    - Es ist immer mehr Funktionalität gefragt! (benutzerdefinierte Datentypen, große Objekte, Multimedia, ...)
  - Gewährleistung hoher Verfügbarkeit
    - Vermeidung eines Systemausfalls
    - Verfügbarkeit aller Daten
    - Mehrrechner-Architekturen erforderlich



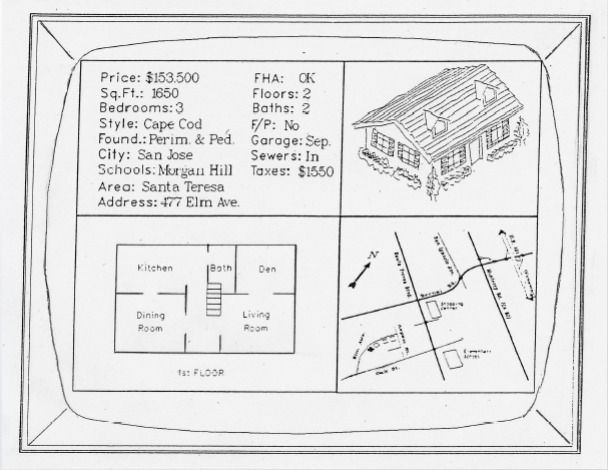
TAS (13)

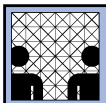
- Anforderungen (Forts.)
  - Modulares Systemwachstum
    - Subsysteme als Einheiten des Entwurfs, der Kontrolle und des Ausfalls
    - Annähernd lineare Durchsatzerhöhung
    - Zuordnung von Prozessoren und Daten (Platten)
  - Einbindung in Client/Server-Architekturen
    - 2-stufig: Präsentation — Anwendung/Datenhaltung
    - 3-stufig: Präsentation — Anwendung — Datenhaltung (Skalierbarkeit!)
    - n-stufig: Einbezug von Web-Servern (und Präsentations-Servern)
- TA-Systeme sind i. allg. horizontal und/oder vertikal verteilt!



TAS (14)

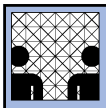
- Anforderungen (Forts.)
  - Beispiel: mehr Funktionalität





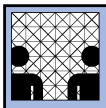
TAS (14)

- Anforderungen (Forts.)
  - Beispiel: mehr Funktionalität (Forts.)
    - Neue Datentypen
      - Vektorgraphik: Grundriss
      - Rasterbilder: Photo des Hauses
      - Video: Tour durchs Haus
    - Neue Funktionen (z. B. DISTANCE_TO, ADJACENT_TO)
  - Solche Anforderungen werden vor allem durch Objekt-Relationale DBS (SQL:1999) und ihre „eingebauten“ Erweiterungsmöglichkeiten erfüllt



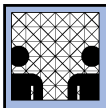
VTAS: Verteilte TAS (1)

- Verteiltes System
  - Es besteht aus autonomen Subsystemen, die oft (weit) entfernt voneinander angeordnet sind, aber koordiniert zusammenarbeiten, um eine gemeinsame Aufgabe zu erfüllen
  - Charakteristisches Kernproblem
    - Mangel an globalem (zentralisiertem) Wissen
    - Lösung im allg. durch fallweise Zuordnung der Kontrolle



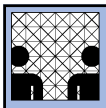
VTAS (2)

- Wirtschaftliche Gründe
  - reduzierte HW-Kosten
  - verfügbare Kommunikationseinrichtungen
- Organisatorische Gründe
  - lokale Unterstützung organisatorischer Strukturen
  - Integration existierender Datenbanken
  - lokale Autonomie
- Technische Gründe
  - Erhöhung der Leistung: Lokalität der Verarbeitung, parallele Verarbeitung
  - größere Verfügbarkeit und Zuverlässigkeit: Minimierung der Auswirkung von „entfernten“ Fehlern, Fehlermaskierung durch Redundanz
  - modulare Wachstumsfähigkeit



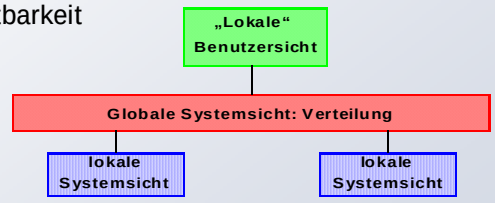
VTAS (3)

- Prinzipien: Funktions-, Daten- und Lastverteilung
- Funktionszuordnung
  - homogener Ansatz: Replikation der Funktionen
  - heterogener Ansatz: Partitionierung von Funktionen, z. B. Client/Server-Architekturen
- Datenzuordnung
  - Partitionierung
  - Replikation
  - Sharing (erfordert lokale Rechneranordnung)
- Lastzuordnung
  - Verteileinheiten durch Funktions- und Datenallokation mitbestimmt: Transaktionsaufträge, Teilprogramme, DB-Operationen (DML-Befehle)



VTAS (4)

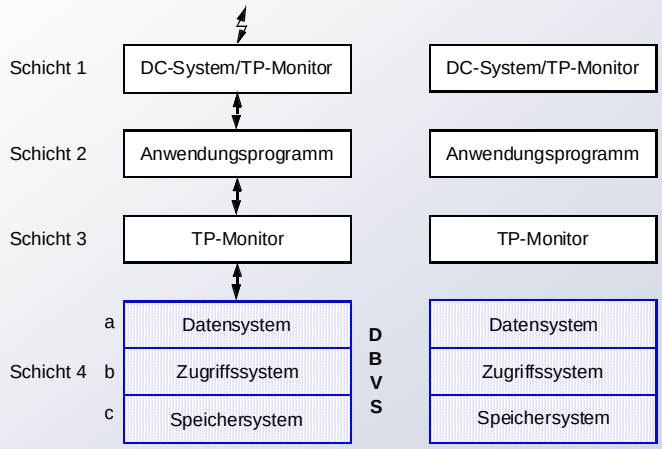
- Vorgehensweise bei der Verteilung
  - Partitionierung der Daten (ggf. mit Replikation)
  - Kommunikation zwischen Prozessen, welche die Zugriffe auf jeweils lokalen Daten ausführen
- Gewünschte Sichtbarkeit



  - Vgl. homogenes Systemmodell auf folgender Folie
  - Forderung: lokale Sicht des Benutzers (Ortstransparenz, „single system image“)
  - Realisierung der globalen Systemsicht: wo?
  - Abbildung auf lokale Sichten der Knoten

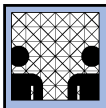
VTAS (5)

- Homogenes Systemmodell



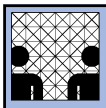
VTAS (6)

- Verteilung unter Kontrolle des TP-Monitors (Verteilte DC-Systeme)
  - TP-Monitor verbirgt weitgehend Heterogenität bezüglich Kommunikationsprotokollen, Netzwerken, BS und Hardware
  - DBS bleiben weitgehend unabhängig (keine Kooperation zwischen DBS)
  - heterogene DBS möglich
    - Einsatz von DBS-Gateways
  - geringe Implementierungskomplexität



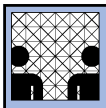
VTAS (7)

- Verteilung unter Kontrolle des TP-Monitors (Forts.)
  - Drei wesentliche Alternativen
    1. ***Transaction Routing***
       - globale Sicht in Schicht 1 (Kommunikationssystem)
       - Einheit der Verteilung ist die Transaktion (TA-Typ)
    2. ***Programmierte Verteilung***
       - globale Sicht in Schicht 2 (im AP)
       - Einheit der Verteilung ist eine Teil-Transaktion (Programmfragment)
    3. ***Verteilung von DB-Operationen (Function Request Shipping)***
       - globale Sicht in Schicht 3 (TP-Monitor)
       - Einheit der Verteilung ist ein DML-Befehl



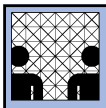
VTAS (8)

- Verteilung unter Kontrolle des TP-Monitors (Forts.)
  - Wesentliche Unterschiede: TAP-Anbindung an DB-Server
    - mehrere Datenquellen – homogen oder heterogen
    - mehrere eigenständige DBMS
    - knotenübergreifende Anwendungsfunktionen wünschenswert
  - Anwendungsanbindung bei VDBS
    - eine logische, lokale DB-Sicht: ein globales DB-Schema
    - eine gleichförmige AP-DB-Schnittstelle (API), z. B. SQL
  - Gleichförmiges API bei heterogenen Datenquellen wünschenswert!



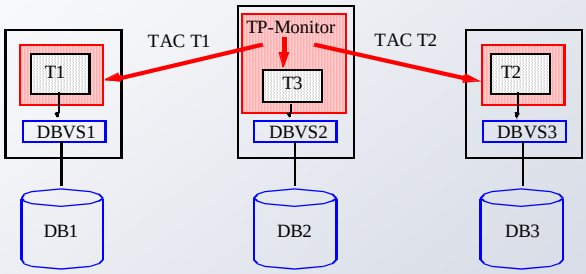
VTAS (9)

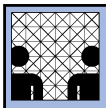
- Transaction Routing
  - Prinzip
    - jeder TA-Typ ist einem Rechner fest zugeordnet
    - TP-Monitor kennt TA-Typ-Zuordnung: Erkennung und Weiterleitung des TAC (→ Ortstransparenz)
    - lokale Transaktionsausführung innerhalb eines Rechners
    - Ausgabenachricht muss ggf. über Zwischenrechner an Benutzer zurückgesendet werden
  - keine Kooperation zwischen DBVS
    - pro Rechner eigene DB/Schemata
    - heterogene DBS möglich
  - als alleiniges Verteilgranulat zu inflexibel (keine echt verteilte Transaktionsausführung)



VTAS (10)

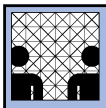
- Transaction Routing (Forts.)





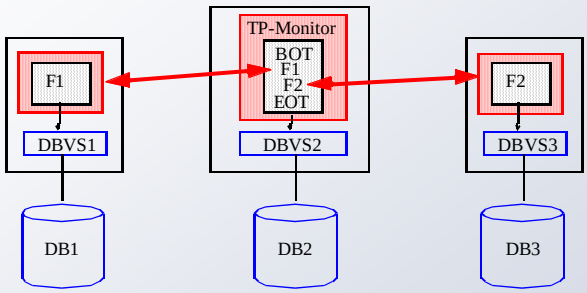
VTAS (11)

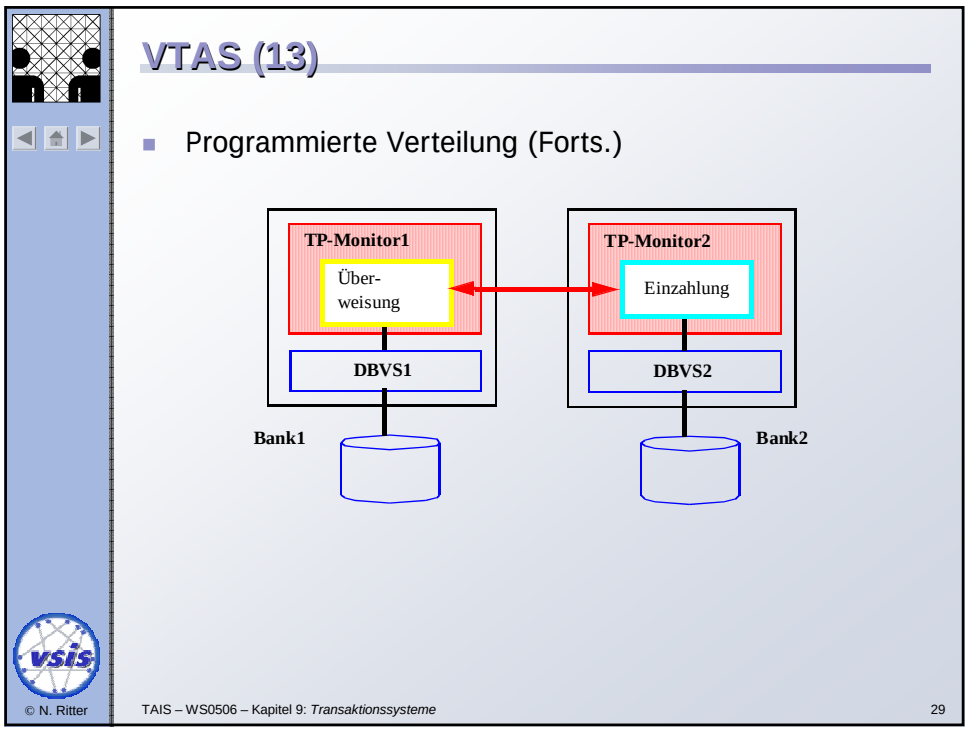
- Programmierte Verteilung
  - Verteilung auf Ebene von Anwendungsprogrammen
    - Zugriff auf externe DB durch Aufruf eines Teilprogramms (RPC) auf dem betreffenden Rechner
    - jedes Teilprogramm greift nur auf lokale DB zu (mit jeweiliger Anfragesprache)
  - Transaktionsverwaltung
    - TP-Monitor koordiniert verteiltes Commit-Protokoll
    - DBS müssen Aufrufe von nicht-lokalen Transaktionen akzeptieren sowie am Commit-Protokoll teilnehmen
    - Auflösung globaler Deadlocks über Timeout
  - Ortstransparenz kann prinzipiell durch TP-Monitor erreicht werden



VTAS (12)

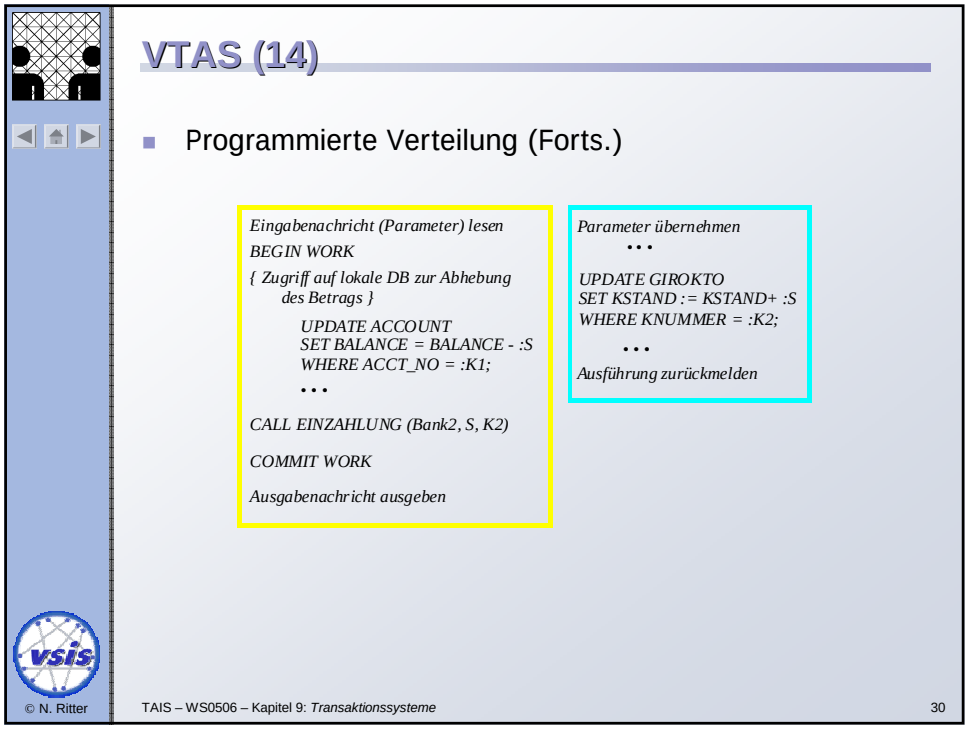
- Programmierte Verteilung (Forts.)





VTAS (13)

- Programmierte Verteilung (Forts.)

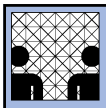


VTAS (14)

- Programmierte Verteilung (Forts.)

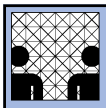
Eingabenachricht (Parameter) lesen
BEGIN WORK
{ Zugriff auf lokale DB zur Abhebung des Betrags }
UPDATE ACCOUNT
SET BALANCE = BALANCE - :S
WHERE ACCT_NO = :K1;
• • •
CALL EINZAHLUNG (Bank2, S, K2)
COMMIT WORK
Ausgabenachricht ausgeben

Parameter übernehmen
• • •
UPDATE GIROKTO
SET KSTAND := KSTAND+ :S
WHERE KNUMMER = :K2;
• • •
Ausführung zurückmelden



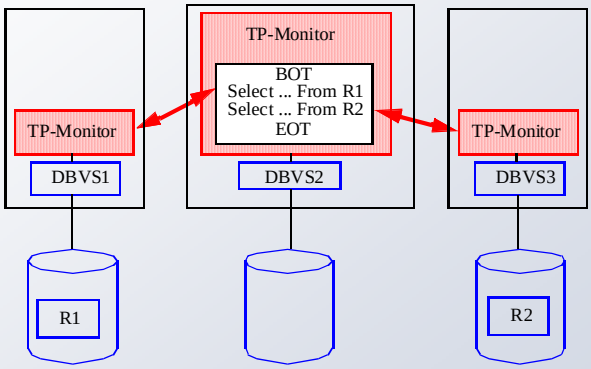
VTAS (15)

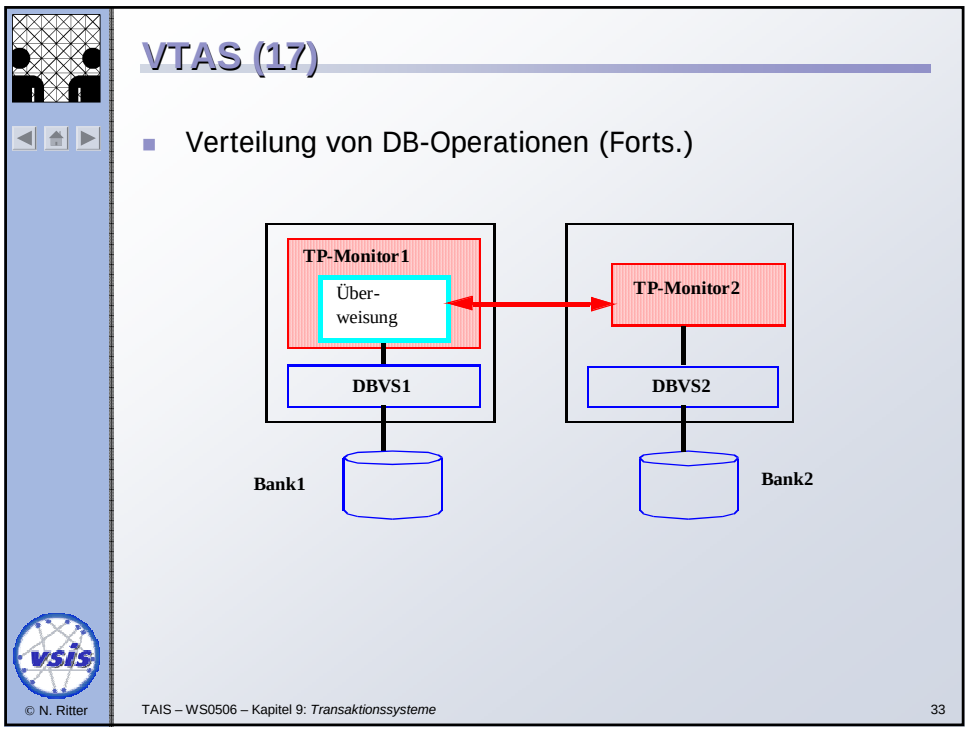
- Verteilung von DB-Operationen
  - DB-Operationen als Verteileinheiten
    - AP können auf entfernte Datenbanken zugreifen
    - Programmierer sieht mehrere Schemata; jede Operation muss innerhalb eines Schemas (DB-Partition) ausführbar sein
    - gemeinsame Anfragesprache wünschenswert
    - Ort der Verteilung kann für AP prinzipiell transparent gehalten werden
    - Transaktionsverwaltung im Prinzip wie bei Programmierter Verteilung



VTAS (16)

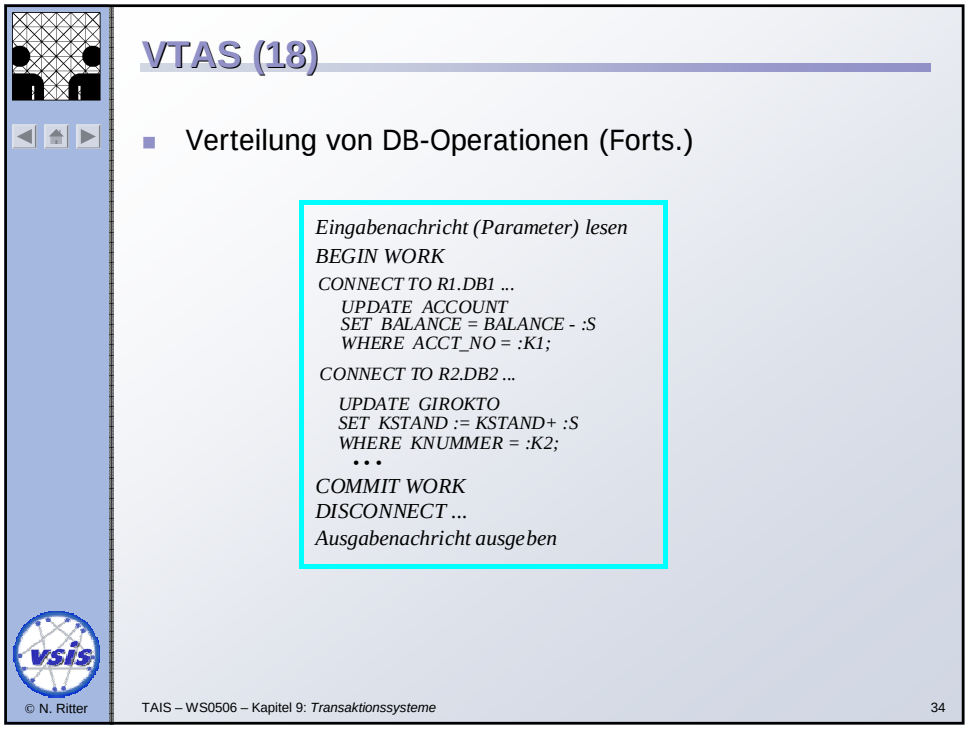
- Verteilung von DB-Operationen (Forts.)





VTAS (17)

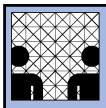
- Verteilung von DB-Operationen (Forts.)



VTAS (18)

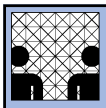
- Verteilung von DB-Operationen (Forts.)

```
Eingabenachricht (Parameter) lesen
BEGIN WORK
CONNECT TO R1.DB1 ...
    UPDATE  ACCOUNT
    SET  BALANCE = BALANCE - :S
    WHERE  ACCT_NO = :K1;
CONNECT TO R2.DB2 ...
    UPDATE  GIROKTO
    SET  KSTAND := KSTAND+ :S
    WHERE  KNUMMER = :K2;
     • • •
COMMIT WORK
DISCONNECT ...
Ausgabenachricht ausgeben
```



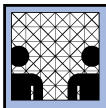
VTAS (19)

- Verteilung unter Kontrolle des TP-Monitors – Bewertung
  - Transaction Routing
    - sehr einfach, ...
    - in heterogenen Umgebungen mit HTTP/HTML
    - sehr starr, keine knotenübergreifenden Transaktionen
  - Programmierte Verteilung
    - sehr hohe Unabhängigkeit
    - Unterstützung von heterogenen DBS
    - geringe Kommunikationshäufigkeit
    - Nutzung bestehender Anwendungsfunktionen möglich
    - gute Administrierbarkeit
    - von vielen Systemen unterstützt
    - geringe Flexibilität des Datenzugriffs (Aufruf bestehender Anwendungsfunktionen, keine Ad-hoc-Anfragen)



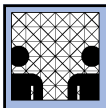
VTAS (20)

- Verteilung unter Kontrolle des TP-Monitors – Bewertung (Forts.)
  - Programmierte Verteilung (Forts.)
    - keine rechnerübergreifenden DB-Operationen
    - geringes Maß an Verteilungstransparenz
  - Verteilung von DB-Operationen
    - größere Freiheitsgrade für DB-Zugriff
    - Verteilungstransparenz, Administrierbarkeit und Kommunikationshäufigkeit schlechter als bei programmierter Verteilung
    - schwierige Programmierung (mehrere DB-Schemata)
    - Bereitstellung einer einheitlichen Programmierschnittstelle (API) für DB-Zugriff und Kommunikation erforderlich



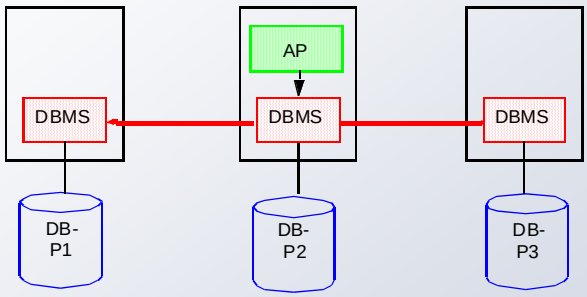
VTAS (21)

- Verteilung unter Kontrolle des DBS
  - → Mehrrechner-DBS
    - vollständige Verteilungstransparenz für TAP
    - setzt eine logische Datenbank voraus
    - geringe Eignung für Knotenautonomie und Heterogenität
  - Mehrere (homogene) Realisierungsalternativen
    - globale Sicht in Schicht 4
    - Verteilung ganzer DML-Befehle bzw. von Teil-Operationen (Schicht 4a)
    - Verteilung von internen Satzoperationen (Schicht 4b)
    - Verteilung von Seitenzugriffen (Schicht 4c)
  - Gleichgestellte Verarbeitungsrechner vs. Spezialisierung (Client/ Server-Konfigurationen)



VTAS (22)

- Mehrrechner-DBS (Forts.)



  - Verteilung unter Kontrolle der DBMS
    - Kooperation zwischen (homogenen) DBMS
    - Ortstransparenz für AP (ein DB-Schema)*: single system image*
    - *Flexibilität für Daten- und Lastverteilung*

VTAS (23)

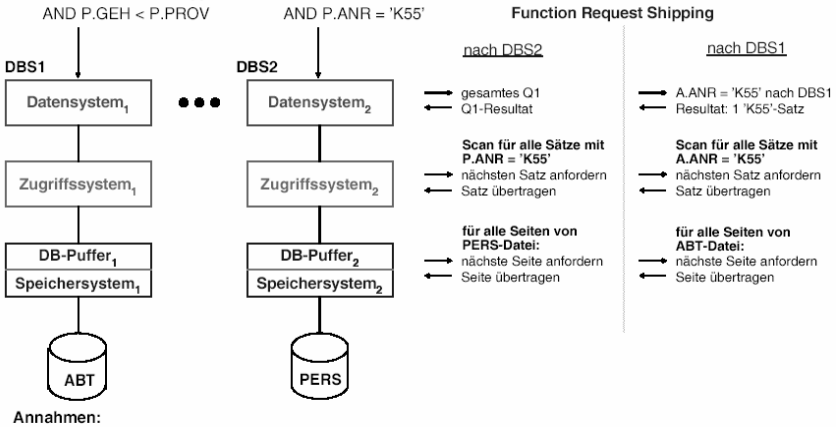
- Mehrrechner-DBS (Forts.)
  - Architekturklassen
    1. DB-Partitionierung (*Shared Nothing*, VDBS)
       - Jeder Knoten besitzt volle Funktionalität und verwaltet eine DB-Partition
       - Datenreplikation erhöht Verfügbarkeit und Zuverlässigkeit
         - Minimierung der Auswirkung von „entfernten“ Fehlern,
         - Fehlermaskierung durch Redundanz
       - Verarbeitungsprinzip: Die Last folgt den Daten
    2. DB-Sharing (*Shared Disk*)
       - Lokale Ausführung von DML-Befehlen
       - Verarbeitungsprinzip: Datentransport zum ausführenden Rechner
       - Lokale Erreichbarkeit der Externspeicher

VTAS (24)

Verarbeitungsaufwand bei VDBS

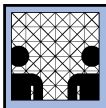
Q_1: SELECT ...
FROM PERS P
WHERE P.ANR = 'K55'
AND P.GEH < P.PROV

Q_2: SELECT ...
FROM PERS P, ABT A
WHERE P.ANR = A.ANR
AND P.ANR = 'K55'



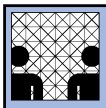
Annahmen:

- PERS-Datei habe 10^5 Seiten
- P.ANR = 'K55': 100 Sätze in PERS
- ABT-Datei habe 10^3 Seiten
- P.ANR = 'K55' AND P.GEH < P.PROV: 5 Sätze in PERS



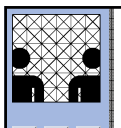
VTAS (25)

- Bereitstellen globaler Sichten bei heterogenen DBS
  - DBS bietet für das Anwendungsprogramm eine vereinheitlichte Sicht auf Daten aus heterogenen Datenquellen (einheitliche Programmierschnittstelle (API))
    - großes Spektrum: DBS, Datei-, Multimedia-, Spreadsheet-, Mail-Systeme
    - Integration einer großen Bandbreite an Datentypen für eine Anwendung möglich
    - Einsatz u. a. bei Legacy-Systemen
    - zwei wesentliche Alternativen:
      1. Schemaabbildung und Anfragetransformation (*schema mapping*, *view translation*)
      2. Middleware-Übersetzer/Optimierer mit Wrapper-Einsatz



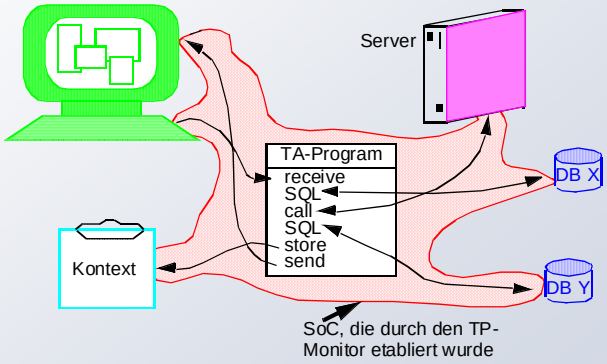
VTAS (26)

- Realisierung allgemeiner Resource-Manager-Architekturen
  - *Resource Manager* (RMs) sind Systemkomponenten, die Transaktionsschutz für ihre gemeinsam nutzbaren Betriebsmittel (BM) bieten
  - Sie gestatten die externe Koordination von BM-Aktualisierungen durch ein 2PC-Protokoll
  - Beispiele: DBS, Dateisysteme, PS-Laufzeitsystem, Mail-Service, Window-Mgr, ...

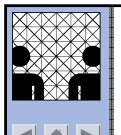


VTAS (27)

- Heterogene Komponenten
  - Integrierte Kontrolle durch Transaktionsdienste – dargestellt in einer typischen Umgebung zur Transaktionsverarbeitung

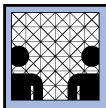


    - Gesamte verteilte Verarbeitung in einer SoC ist eine ACID-Transaktion
      - Alle Komponenten werden durch die TA-Dienste integriert
      - Für die Kooperation ist eine Grundmenge von Protokollen erforderlich



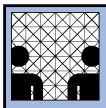
VTAS (28)

- Heterogene Komponenten (Forts.)
  - Transaktionsschutz durch kooperierende *Resource Manager*
    - Gewährleistung der ACID-Eigenschaften für DB-Daten und auch für andere Betriebsmittel (persistente Warteschlangen, Nachrichten, Objekte von persistenten Programmiersprachen)
    - Globaler Transaktionsschutz durch Resource-Mgr-Architektur unter Kontrolle eines TP-Monitors



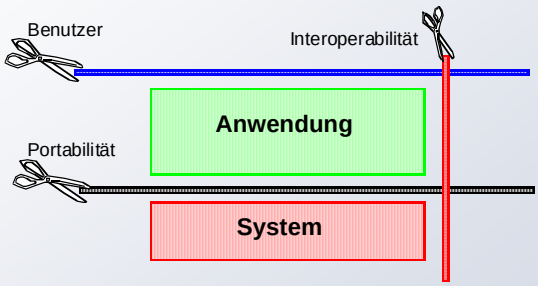
TA-Verarbeitung in offenen Systemen (1)

- Idee der Client/Server-Verarbeitung korrespondiert mit dem Konzept von Offenen Systemen.
- Definition von IEEE im Rahmen der POSIX-Aktivität
  - Ein *offenes System* ist
    - „ein System, das in ausreichendem Maße offengelegte Spezifikationen für Schnittstellen und dazugehörige Formate implementiert, damit entsprechend gestaltete Anwendungssoftware
      - auf eine Vielzahl verschiedener Systeme portiert werden kann (mit Anpassungen),
      - mit anderen Anwendungen lokal und entfernt interoperabel ist,
      - mit Benutzern in einer Art interagiert, die das Wechseln der Benutzer zwischen Systemen erleichtert“.
    - Diese Anforderungen spiegeln sich z. T. auch in den Zielvorstellungen von Client/Server-Systemen wider.

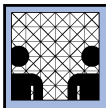


TA-Verarbeitung in offenen Systemen (2)

- Offenheit impliziert Standardisierung
  - Definition und Veröffentlichung von Schnittstellen

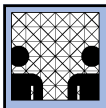


  - Systemschnittstellen gewährleisten die Portabilität der AW-Software
  - Aufrufschnittstellen erlauben die Interoperabilität zwischen AWs und zwischen Systemen
  - Benutzerschnittstellen regeln einen einheitlichen Benutzerzugang



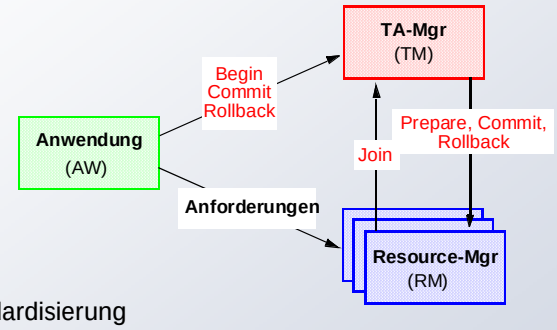
TA-Verarbeitung in offenen Systemen (3)

- Standards zur TA-Verwaltung
  - TP-Monitore benötigen Standards, weil sie letztendlich den „Klebstoff“ verkörpern, der die unterschiedlichen und heterogenen SW-Komponenten zusammenfügt
  - ihre AW laufen auf verschiedenartigen Plattformen und
  - haben Zugriff zu verschiedenen DBs und RMs
  - die AW haben absolut kein Wissen voneinander
  - einziger Weg zur Verknüpfung: „Offene Standards“ Bereitstellung von Schnittstellen zwischen TP-Monitor und RMs, TP-Monitor untereinander und TP-Monitor und AW
- Standards kommen aus zwei Quellen
  - **ISO: OSI-TP** spezifiziert die Nachrichtenprotokolle zur Kooperation von TP-Monitoren
  - X/OPEN definiert den allgemeinen Rahmen für TA-Verarbeitung **X/Open DTP** (distributed transaction processing) stellt eine SW-Architektur dar, die mehreren AWP erlaubt, gemeinsam Betriebsmittel mehrerer RMs zu nutzen und die Arbeit der beteiligten RMs durch globale Transaktionen zusammenzufassen.

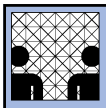


TA-Verarbeitung in offenen Systemen (4)

- X/Open DTP (1991) für die lokale Zusammenarbeit von Anwendung (AW), TA-Mgr (TM) und Resource-Mgrs (RMs)

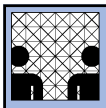


- Standardisierung
  - Schnittstelle zur Transaktionsverwaltung (TX: AW — TM)
  - Schnittstelle zwischen TM und RMs
    (XA: zur TA-Verwaltung und ACID-Kontrolle)
  - zusätzlich: Anforderungsschnittstellen (API, z. B. SQL)



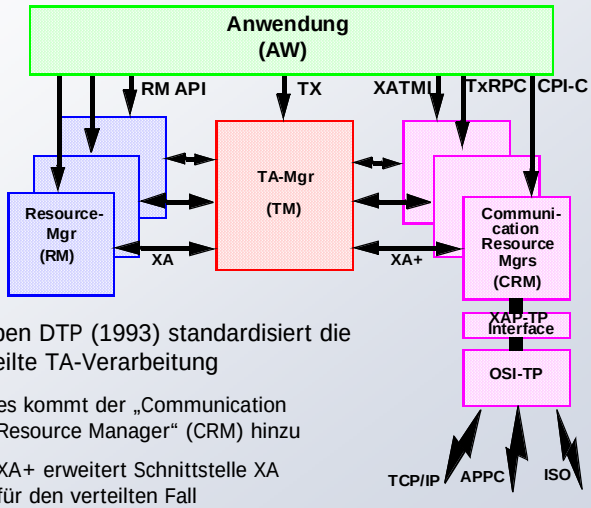
TA-Verarbeitung in offenen Systemen (5)

- X/Open DTP (Forts.)
  - TA-Ablauf
    - die AW startet eine TA, die vom lokalen TA-Mgr verwaltet wird
    - RMs melden sich bei erster Dienst-Anforderung beim lokalen TA-Mgr an (Join)
    - wenn AW Commit oder Rollback ausführt (oder zwangsweise zurück-gesetzt wird), koordiniert der TA-Mgr und schickt Ergebnis zu den RMs (über Broadcast)
    - hier sorgen die RMs für Synchronisation und Logging in ihrem Bereich (private Lock-Mgr und Log-Mgr)

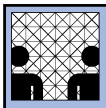


TA-Verarbeitung in offenen Systemen (6)

- X/Open DTP (Forts.)

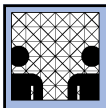


  - X/Open DTP (1993) standardisiert die verteilte TA-Verarbeitung
    - es kommt der „Communication Resource Manager“ (CRM) hinzu
    - XA+ erweitert Schnittstelle XA für den verteilten Fall



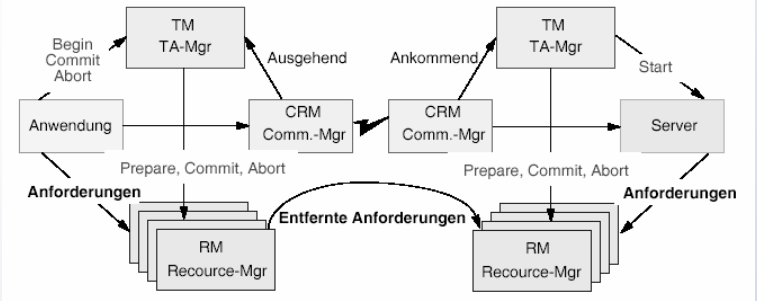
TA-Verarbeitung in offenen Systemen (7)

- Definition von Schnittstellen auf der AW-Ebene
  - TxRPC definiert transaktionalen RPC; seine TA-Eigenschaften können ausgeschaltet werden (optional)
  - CPI-C V2 ist Konversationsschnittstelle (peer-to-peer), welche die OSI-TP-Semantik unterstützen soll
  - XATMI ist Konversationsschnittstelle für Client/Server-Beziehungen
- Sind drei Schnittstellen-Standards erforderlich?
  - Kompromisslösung für drei existierende Protokolle (DCE, CiCS, Tuxedo)



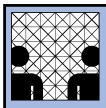
TA-Verarbeitung in offenen Systemen (8)

- Definition von Schnittstellen auf der AW-Ebene



TA-Ablauf

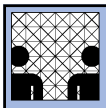
- AW startet TA, die vom lokalen TA-Mgr verwaltet wird
- Wenn die AW oder der RM, der für die AW eine Anforderung bearbeitet, eine entfernte Anforderung durchführen, informieren die CRMs an jedem Knoten ihre lokalen TA-Mgr über die ankommende oder ausgehende TA
- TA-Mgr verwalten an jedem Knoten jeweils die TA-Arbeit am betreffenden Knoten
- Wenn die AW COMMIT oder ROLLBACK durchführt oder scheitert, kooperieren alle beteiligten TA-Mgr, um ein atomares und dauerhaftes Commit zu erzielen.



TP-Monitor (1)

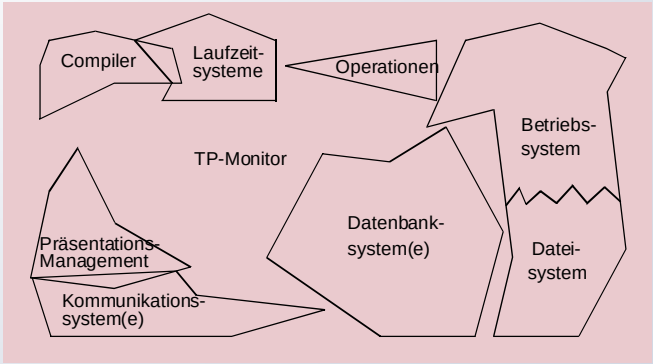
- TP-Monitore (Transaction Processing Monitor)
  - bieten schon seit ~ 1970 auf Mainframes robuste Laufzeitumgebungen für große OLTP-Anwendungen *(on-line transaction processing)*
  - liefern den „Klebstoff“ für das Zusammenwirken vieler Komponenten, Betriebsmittel (BM), Protokolle usw. bei der TA-Abwicklung
  - realisieren und optimieren Funktionen, die von BS typischerweise nur sehr schlecht oder gar nicht unterstützt werden
  - verwalten Prozesse und starten/überwachen TAPs; sie erlauben die Integration unabhängiger Dienste und ihre Abwicklung als TAs

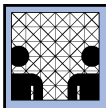
“In a contest for the least well-defined software term, TP-Monitor would be a tough contender” (J. Gray)



TP-Monitor (2)

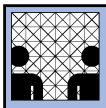
- TP-Monitor und zugeordnete Systemkomponenten
  - Der TP-Monitor integriert verschiedenartige Systemkomponenten, um gleichförmige Schnittstelle für Anwendungen und Operationen mit demselben Verhalten im Fehlerfall *(failure semantics)* zu bieten.





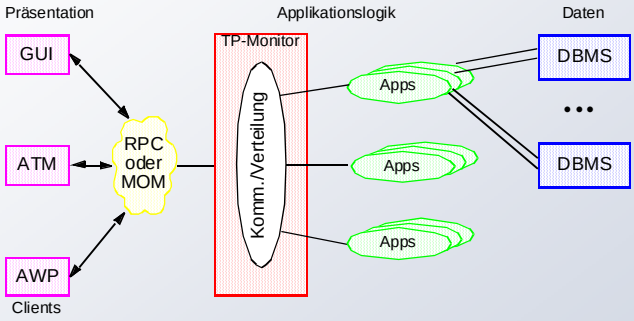
TP-Monitor (3)

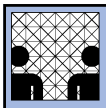
- Neue TP-Monitor-Generation
  - *TP-Monitors are like the Rolling Stones – been around for a long time, but still drawing large crowds* (David Linthicum)
  - ist „offen“ (standardisierte Schnittstellen zu Plattformen, Kommunikation, Benutzern)
  - lässt sich auf mehreren BS (Rechnerplattformen) einsetzen
  - ist Client-/Server-basiert
- Was tun TP-Monitore in C/S-Umgebungen?
  - Sie verwalten Transaktionen und koordinieren ihren Ablauf im System
  - Sie kontrollieren die Kommunikationsflüsse zwischen Tausenden von Clients und Hunderten von Servern
  - Sie ergreifen Maßnahmen zur Lastbalancierung und verbessern das Leistungsverhalten
  - Sie sind für den Wiederanlauf nach Fehlern *(Restart)* verantwortlich
  - Sie stellen sicher, dass Transaktionen die ACID-Eigenschaften einhalten



TP-Monitor (4)

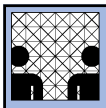
- Was tun TP-Monitore in C/S-Umgebungen? (Forts.)





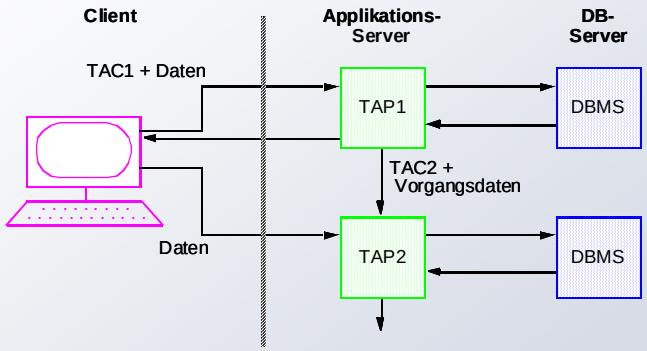
TP-Monitor (5)

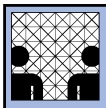
- Wie sind TP-Monitor-basierte Anwendungen aufgebaut?
  - TP-Monitor bietet ein vordefiniertes Framework für Entwicklung, Betrieb und Administration von C/S-Applikationen
  - auf der Client-Seite sind Tools zur Entwicklung/Definition von GUIs (Fenster, Formulare, Masken usw.) verfügbar
  - auf der Server-Seite lassen sich modulare, wiederbenutzbare Dienste entwickeln, die von Ressourcen-Mgr (RM) gekapselt werden
  - TP-Monitore stellen allgemeine Server-Klassen zur Verfügung, für die Prozesse erzeugt werden können, in denen die Dienste der Applikation abgewickelt werden (ein oder mehrere Prozesse pro Server-Klasse)
  - Sie führen auf Server-Seite einen ereignisgetriebenen Programmierstil ein (TACs initiieren TAPs)



TP-Monitor (6)

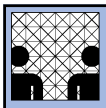
- Prinzipieller Ablauf eines Vorgangs





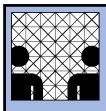
TP-Monitor (7)

- Was ist nun genau ein TP-Monitor?
  - Er lässt sich als BS für transaktionsgeschützte Applikationen auffassen
  - Er ist ein Framework für Applikations-Server
  - Er erledigt drei Aufgaben extrem gut
    - Prozessverwaltung
      - das Starten von Server-Prozessen,
      - das Initiieren von TAPs,
      - das Kontrollieren ihres Ablaufs
      - die Lastbalancierung



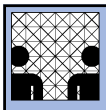
TP-Monitor (8)

- Was ist nun genau ein TP-Monitor? (Forts.)
  - Er erledigt drei Aufgaben extrem gut (Forts.)
    - Transaktionsverwaltung
      - sie garantiert die ACID-Eigenschaften von allen Programmen, die unter ihrem „Schutz“ ablaufen
      - dazu muss sie im Normalbetrieb Logging (z. B. Protokollieren von Nachrichten) durchführen, um im Fehlerfall Recovery-Maßnahmen ergreifen zu können
      - *The Standish Group estimates that the world electronically processes 68 million transactions every second. 53 million of the 68 million use a TP Monitor.* (Jim Johnson, Oct. 1998)



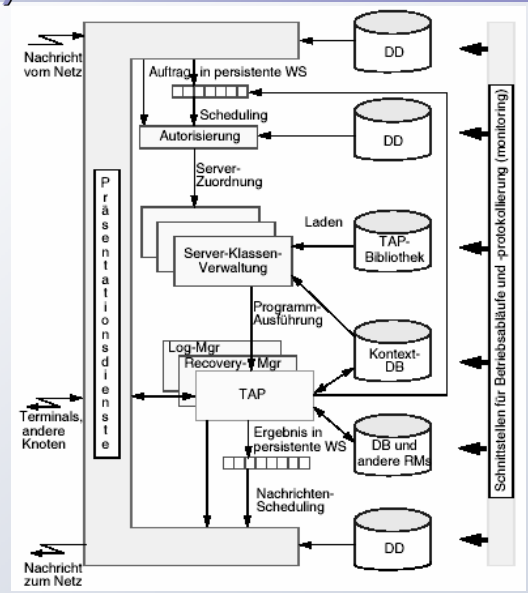
TP-Monitor (9)

- Was ist nun genau ein TP-Monitor? (Forts.)
  - Er erledigt drei Aufgaben extrem gut (Forts.)
    - C/S-Kommunikationsverwaltung
      - es ist Client/Server- und Server/Server-Kommunikation zu unterstützen
      - sie erlaubt den Client- und Server-Programmen, die an einer Anwendung beteiligten Dienste und Komponenten auf verschiedene Weise aufzurufen: RPCs, Konversationen, asynchrone Nachrichten über persistente Warteschlangen (MOM: *message-oriented middleware*), Broadcasts, ...

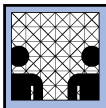


TP-Monitor (10)

- Kontrollfluss durch einen TP-Monitor (vereinfacht)

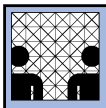


- Diagramm repräsentiert die wichtigsten Grundfunktionen für TAP-Verwaltung und -Ausführung



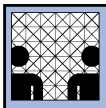
TP-Monitor (11)

- Präsentationsdienste
  - bilden die Schnittstellen zwischen dem TAP und den E/A-Geräten
  - bieten Geräteunabhängigkeit (Terminaltyp, Formatkontrolle, *Scrolling*) und Kommunikationsprotokollunabhängigkeit
  - wegen vieler verschiedener Präsentationsdienste (Window-Protokolle, Graphikstandards) implementieren oft eigene RM diese Dienste
- Warteschlangenverwaltung
  - WS-Dienste müssen transaktionsorientiert sein: Auftrag in WS wird genau einmal ausgeführt ('exactly once')
  - nur bei Commit der Server-TA kommt Ergebnis in Ausgabe-WS
  - abhängig von Anwendung und Inhalt der Nachricht können für Auslieferung verschiedene Zusicherungen vereinbart werden: 'at least once', 'at most once' oder 'exactly once'
  - WS-Mgr ist oft als RM implementiert, der zum TP-Monitor gehört



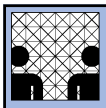
TP-Monitor (12)

- Server-Klassen-Verwaltung
  - zuständig dafür, dass für jedes TAP eine Server-Klasse definiert und aktiv ist
  - Aktivierung der Server entweder 'by default' oder 'on demand'
  - Aufgaben: Erzeugen von Prozessen und WS, Laden der TAP-Codes, Besorgen der Zugriffsrechte für die WS-Zugriffe, Festlegen der Server-Prioritäten u.a.
  - Aufgaben fallen auch in die Verantwortlichkeit der Lastbalancierung; sie sind deshalb in enger Kooperation zu lösen
  - Funktionen wie Prozesserzeugung und TAP-Ausführung werden vom BS zur Verfügung gestellt



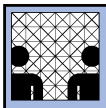
TP-Monitor (13)

- Scheduling von Aufträgen
  - die am häufigsten angeforderte Funktion des TP-Monitors
  - Bestimmung des angeforderten Dienstes: lokal oder entfernter Knoten
  - Weiterleitung des Auftrags an den Server
- Autorisierung der Aufträge
  - Teil der systemweiten Sicherheitsvorkehrungen
  - Spektrum der Lösungen: von einfacher, statischer Autorisierung bis zu wertabhängiger, dynamischer Autorisierung
  - wegen spezifischer Betriebscharakteristika von TA-Systemen ist dynamische Autorisierung für individuelle Aufträge sehr wichtig



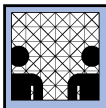
TP-Monitor (14)

- Kontextverwaltung
  1. Speicherung von Verarbeitungskontexten, die über TA-Grenzen gehalten werden sollen (Mehr-TA-Vorgänge)
     - Kontext-DB hat alle ACID-Eigenschaften und könnte durch ein SQL-DBS implementiert werden
  2. Unterstützung der Kontextnutzung innerhalb einer TA: Weitergabe von Zwischenergebnissen einer TA über Nachricht beim Server-Aufruf zu aufwendig; Kontextverwaltung speichert die Daten zwischen und erlaubt nachfolgenden Servern Zugriff auf die Daten
     - diese Zugriffe erfolgen innerhalb einer TA; deshalb sind keine persistenten Kontexte erforderlich



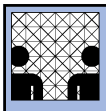
TP-Monitor (15)

- Metadatenverwaltung
  - Annahme: globales Repository (DD, Data Dictionary)
  - je mehr der TP-Monitor tun soll, desto genauer/umfangreicher müssen die Metadaten sein
  - Metadaten-Übersicht: Beschreibung der
    - zum verteilten TA-System gehörigen Knoten: Namen, Adressen, ...
    - lokalen Komponenten der TA-Dienste wie Log-Mgr, Recovery-Mgr, Kommunikations-Mgr, ...
    - Geräte und HW-Komponenten, die der TP-Monitor kennen muss, wie Terminals, Controller, physische Verbindungen, ...
    - TAPs und RMs, die am betreffenden Knoten installiert sind
    - Autorisierungsinformation
      - Zugriffskontrollisten für TAPs und RMs
      - über die dem System bekannten Benutzer
      - Autorisierungs-Codes (Passwörter), Sicherheitsprofile, ...



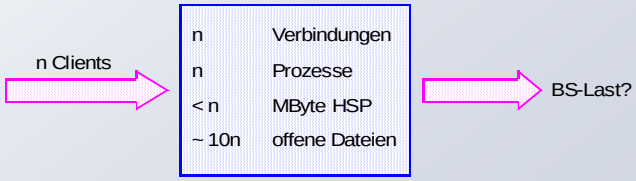
TP-Monitor (16)

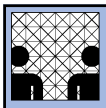
- Metadatenverwaltung (Forts.)
  - Metadaten-Übersicht: Beschreibung der (Forts.)
    - Konfigurationsdaten
      - für Server-Klassen über Prozesse, Tasks, Prioritäten
      - über Betriebs- und Administrations-Schnittstellen
      - für Wiederanlauf und spezielle Abläufe (Restart-Reihenfolge der RMs)
    - Inhalt dieser Kataloge wird gewartet und aktualisiert über System-TAs
    - beim Restart wird die hier hinterlegte Konfiguration „hochgefahren"



TP-Monitor (17)

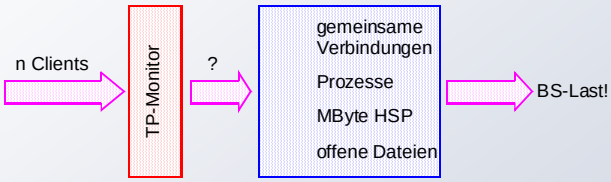
- Ablauf in 3-stufigen C/S-Umgebungen
  - Typischer BM-Bedarf pro Client auf einem Server
    - eine Kommunikationsverbindung
    - 0.5 — 1 MByte Hauptspeicher (RAM)
    - 1 oder 2 Prozesse
    - ~ 10 offene Dateikontrollblöcke *(file handle)*
    - Soll jeder Client seine eigenen BM statisch zugeordnet bekommen (virtueller Prozessor)?
  - Herkömmliche BS-Abbildung



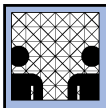


TP-Monitor (18)

- Ablauf in 3-stufigen C/S-Umgebungen (Forts.)
  - Einsatz eines TP-Monitors

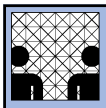


    - "TP-Monitor: The 3-Tier Workhorse" (Jeri Edwards)



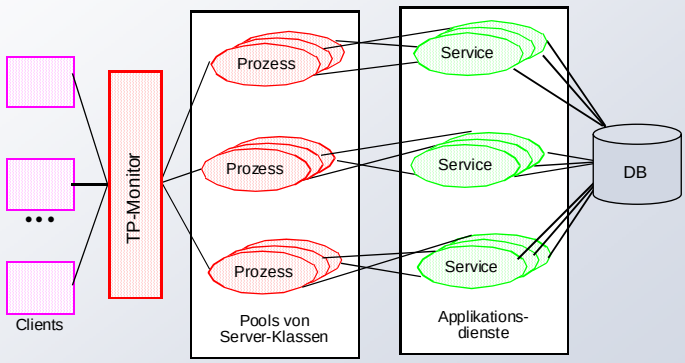
TP-Monitor (19)

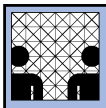
- Ablauf in 3-stufigen C/S-Umgebungen (Forts.)
  - BM-Zuteilung erfolgt auftragsbezogen
    - OLTP-Applikation besteht aus einer Menge von Diensten (TAPs)
    - Server-Klasse besteht aus Pool von (statisch erzeugten) Prozessen (mit Tasks/Threads), welche vorab geladene TAPs abwickeln können; TP-Monitor kann dynamisch neue Prozesse starten.
    - Lastbalancierung
    - Applikation kann eine oder mehrere Server-Klassen besitzen
    - TP-Monitor weist Server-Klasse ankommenden Auftrag (TAC) zu; nach Abwicklung Freigabe der auftragsbezogenen BM leitet TP-Monitor die Antwort an den Client weiter



TP-Monitor (20)

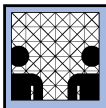
- Ablauf in 3-stufigen C/S-Umgebungen (Forts.)





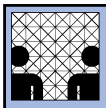

TP-Monitor (21)

- Ablauf in 3-stufigen C/S-Umgebungen (Forts.)
  - Bildung von Server-Klassen
    - nach Prioritätsaspekten für die TAP-Ausführung
    - nach Applikationstypen
    - nach Antwortzeitvorgaben
    - nach Fehlertoleranzanforderungen, ...




Benchmarks (1)

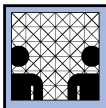
- Benchmark: Vergleichspunkt, Bezugswert, Maßstab
- Antwortzeit
  - ist bei interaktiven TA ist eine kritische Größe; Stapel-TA können dagegen Stunden oder Tage laufen und enorme Ressourcen verbrauchen (CPU, Speicher)
- Leistungsbestimmung eines TA-, DW-, DSS-Systems ist sehr schwierig
  - Komplexität des Systems
  - Leistungsverhalten verschiedenartiger Lasten oft sehr verschieden!
- Gesucht: Test zur Evaluation der wesentlichen Leistungsmerkmale
  - Anforderungen spezifiziert als anwendungsbezogene Funktionalität (TA-Typen)
  - Aussagen über gesamte Systemkosten
  - Leistungsverhalten bei Wachstum der TA-Last und/oder der Datenvolumina (Skalierbarkeit)
  - objektive Leistungsmaße wie Durchsatz und Antwortzeit zur einfachen Vergleichbarkeit (bei Systemwachstum oder konkurrierenden Systemen)



Benchmarks (2)

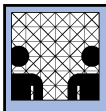
- Typische Leistungsmaße/-werte interaktiver TA

Leistung/TA	einfach	mittel	komplex
Instr./TA	100 K	1 M	100 M
Platten-E/A/TA	1	10	1000
Lokale Nachr. (B)	10 (5KB)	100 (50KB)	1000 (1MB)
Entfernte Nachr. (B)	2 (300B)	2 (4KB)	100 (1MB)
Kosten/TA/sec.	< 0.1K$/tps	< 1K$/tps	<10K$/tps
Spitzen-TPS/Knoten	> 1000	> 100	> 1



Benchmarks (3)

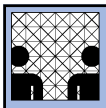
- Im Laufe der Zeit wurden unterstützende Verfahren entwickelt
  - Es wurden verschiedene Nutzungsmuster von Anwendungen erkannt, die als Benchmark nachgebildet wurden
  - Probleme früher Benchmarks
    - SUT *(system under test)* lieferte nichtssagende Leistungszahlen, da Netz- und Benutzerinteraktion vernachlässigt wurden
    - Sie waren unvollständig spezifiziert und wurden nicht überwacht (durch “Notar”)
    - Marketing-Aussagen waren oft unglaubwürdig!
- Ziele
  - Definition von aussagefähigen Benchmarks, um typische TA-Lasten bestimmter Anwendungsbereiche zu repräsentieren
  - Spezifikation eines durchgängigen Prozesses zur
    - Anpassung, Ergänzung und Überwachung von Benchmarks
    - Gewährleistung der Vergleichbarkeit der Messungen (und damit der Systeme)



Benchmarks (4)

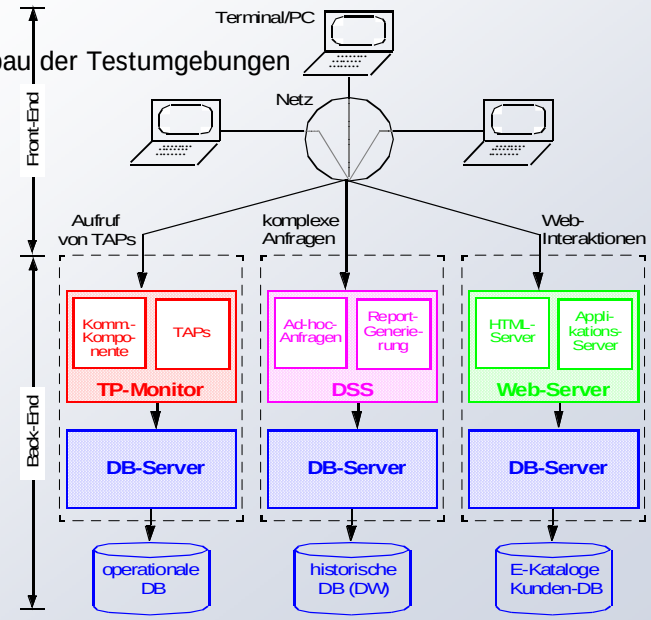
- Standardisierung durch TPC
  (Transaction Processing Performance Council)
  - Konsortium aus z. Zt. 41 Organisationen
    (HP, IBM, Microsoft, Sun, ...)
  - Formelles Verfahren zur Standardisierung von Benchmarks:
    TPC-A (1989), TPC-B (1990), TPC-C (1992), TPC-D (1994), ...
  - Unabhängige Instanzen zur Überwachung
- "Good benchmarks are like good laws"

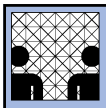
Gray (ed.): The Benchmark Handbook for Database and Transaction Processing Systems. Morgan Kaufmann 1991



Benchmarks (5)

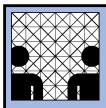
- Grobaufbau der Testumgebungen





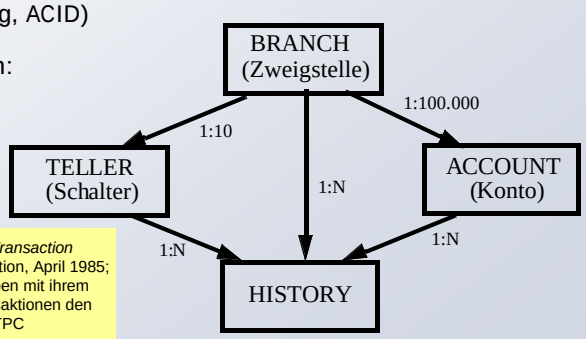
Benchmarks (6)

- Grobaufbau der Testumgebungen (Forts.)
  - Optionen für SUT (system under test): Messung der Aktionen
    - im Gesamtsystem (End-to-End)
    - nur im Server (Back-End)
  - Struktur des Back-End
    - zentralisierte Anordnung oder
    - verteilte Komponenten (Client/Server-Architektur)
    - m Anwendungs-, n DB-Server-Instanzen

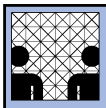


Benchmarks (7)

- TPC-A: erste Standardisierung durch TPC
  - Standard-Transaktion ‚Kontenbuchung' (DebitCredit)
  - basierend auf dem IBM-Benchmark TP1 (~1985), der die Systemleistung bei Abwicklung von ATM-Transaktionen (*Automated Teller Machine*) ohne Netz- und Interaktionskomponente (Denkzeit) erfasste
  - Vervollständigung der (TP1-)Definition (Datenstrukturen, Skalierung, ACID)
  - Satztypen:

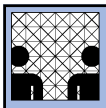


Anon et al.: *A Measure of Transaction Processing Power*, Datamation, April 1985; J. Gray und 24 Autoren gaben mit ihrem Artikel zu DebitCredit-Transaktionen den Anstoß zur Gründung des TPC



Benchmarks (8)

- TPC-A (Forts.)
  - Mengengerüst abhängig vom Durchsatzziel (in TA pro Sekunde (tps))
    - pro tps:
      - 1 BRANCH-Satz
      - 10 TELLER-Sätze
      - 100.000 ACCOUNT-Sätze
      - HISTORY für 90 Tage (90*8*60*60 = 2.592.000 Sätze)
    - daneben 10 Terminals pro tps



Benchmarks (9)

- TPC-A (Forts.)
  - Transaktionsprogramm

Read message from Terminal *(acctno, branchno, tellerno, delta)*;
BEGIN_WORK { Beginn der Transaktion }

UPDATE ACCOUNT
SET balance = balance + *:delta*
WHERE acct_no = *:acctno*

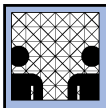
SELECT balance INTO *:abalance*
FROM ACCOUNT
WHERE acct_no = *:acctno;*

UPDATE TELLER
SET balance = balance + *:delta*
WHERE teller_no = *:tellerno*

UPDATE BRANCH
SET balance = balance + *:delta*
WHERE branch_no = *:branchno*

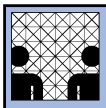
INSERT INTO HISTORY (Tid, Bid, Aid, delta, time)
VALUES (*:tellerno, :branchno, :acctno, :delta,* CURRENT);

COMMIT_WORK ; { Ende der Transaktion }
Write message to Terminal (*abalance*, ...);



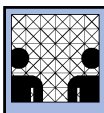
Benchmarks (10)

- TPC-A (Forts.)
  - Benchmark-Forderungen:
    - 15% der Transaktionen betreffen Konto einer anderen Zweigstelle
    - 90% der TA sollen Antwortzeit von höchstens 2 Sekunden haben
  - Bestimmung der Kosten $/tps:
    - Systemkosten umfassen HW-, SW- und Kommunikations-Komponenten, Terminals, Backup-Speicher sowie Wartung für 5 Jahre
    - Systemkosten ($) / Durchsatz pro Sekunde (tps)
  - Unterscheidung von geographisch und lokal verteilten Systemen
- TPC-B:
  - keine Berücksichtigung des Netzes und der Terminals (Back-End)
  - Kritik: tps künstlich hoch, $/tps künstlich niedrig



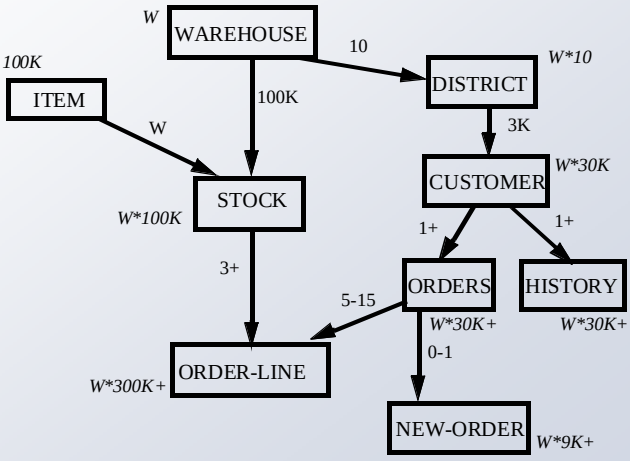
Benchmarks (11)

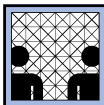
- TPC-C
  - Modellierung/Abwicklung von Aktivitäten im Großhandel
    - repräsentativ für komplexe OLTP-Anwendungsumgebungen
    - Verwaltung, Verkauf, Verteilung von Produkten oder Dienstleistungen
    - Unternehmen besitzt geographisch verteilte Geschäftsdistrikte und zugeordnete Filialen
    - realistischere Transaktionslast aus mehreren Transaktionstypen unterschiedlicher Komplexität und Änderungshäufigkeit
  - Anwendung: Bestellverwaltung im Großhandel
    - Betrieb umfasst W Warenhäuser, pro Warenhaus 10 Distrikte, pro Distrikt 3000 Kunden
    - 100.000 Artikel; pro Warenhaus wird Anzahl vorhandener Artikel geführt
    - 1% aller Bestellungen werden von nicht-lokalem Warenhaus angefordert



Benchmarks (12)

- TPC-C (Forts.)
  - 9 Satztypen





Benchmarks (13)

- TPC-C (Forts.)
  - Haupt-Transaktionstyp: New-Order

BEGIN_WORK { Beginn der Transaktion }

SELECT ... FROM **CUSTOMER**
WHERE c_w_id = *:w_no* AND c_d_id = *:d_no* AND c_id = *:cust_no*

SELECT ... FROM **WAREHOUSE**
WHERE w_id = *:w_no*

SELECT ... FROM **DISTRICT** (* -> *next_o_id* *)
WHERE d_w_id = *:w_no* AND d_id = *:d_no*

UPDATE DISTRICT
SET d_next_o_id := *:next_o_id* + 1
WHERE d_w_id = *:w_no* AND d_id = *:d_no*

INSERT INTO NEW_ORDER ...

INSERT INTO ORDERS ...

pro Artikel (im Mittel 10) werden folgende Anweisungen ausgeführt:

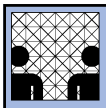
SELECT ... FROM **ITEM** WHERE ...

SELECT ... FROM **STOCK** WHERE ...

UPDATE STOCK ...

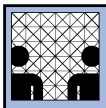
INSERT INTO ORDER-LINE ...

COMMIT_WORK { Ende der Transaktion }



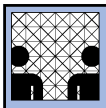
Benchmarks (14)

- TPC-C (Forts.)
  - Haupt-Transaktionstyp: New-Order (Forts.)
    - im Mittel 48 SQL-Anweisungen
      (BOT, 23 SELECT, 11 UPDATE, 12 INSERT, EOT)
    - Durchsatzangabe für New-Order-Transaktionen in tpmC
      (Transaktionen pro Minute)
  - Transaktionstypen:
    - *New-Order:* Artikelbestellung (Read-Write)
    - *Payment:* Bezahlung einer Bestellung (Read-Write)
    - *Order-Status*: Status der letzten Bestellung eines Kunden ausgeben (Read-Only)
    - *Delivery:* Verarbeitung von 10 Bestellungen (Read-Write)
    - *Stock-Level:* Anzahl von verkauften Artikeln bestimmen, deren Bestand unter bestimmtem Grenzwert liegt (Read-Only)



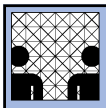
Benchmarks (15)

- TPC-C (Forts.)
  - Festlegung des Transaktionsmixes
    - Payment-TA müssen mindestens 43% der Last ausmachen
    - Order-Status, Delivery und Stock-Level je mindestens 4%
    - New-Order-Anteil ist variabel; solange es die Antwortzeitrestriktionen zulassen, werden bei einer Messung diese Transaktionen generiert
  - Antwortzeitrestriktionen
    - 90% unter 5 Sek. (New-Order, Payment, Order-Status, Delivery)
    - Stock-Level: 90% unter 20 Sek.
  - Was bedeutet tpmC?
    - Es wird nur der Durchsatz von New-Order-Transaktionen pro Minute gemessen, wobei das System parallel die restlichen 4 Transaktionstypen unter Beachtung der Restriktionen des Transaktionsmixes und der Antwortzeit ausführt
    - SUT-Messung: End-to-End
  - $/tpmC: Systemkosten ($) / Durchsatz pro Minute (tpmC)



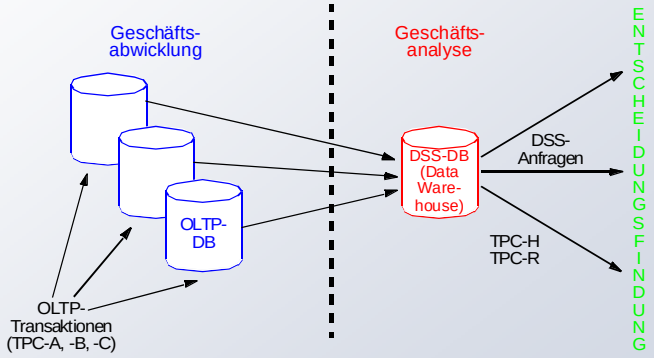
Benchmarks (16)

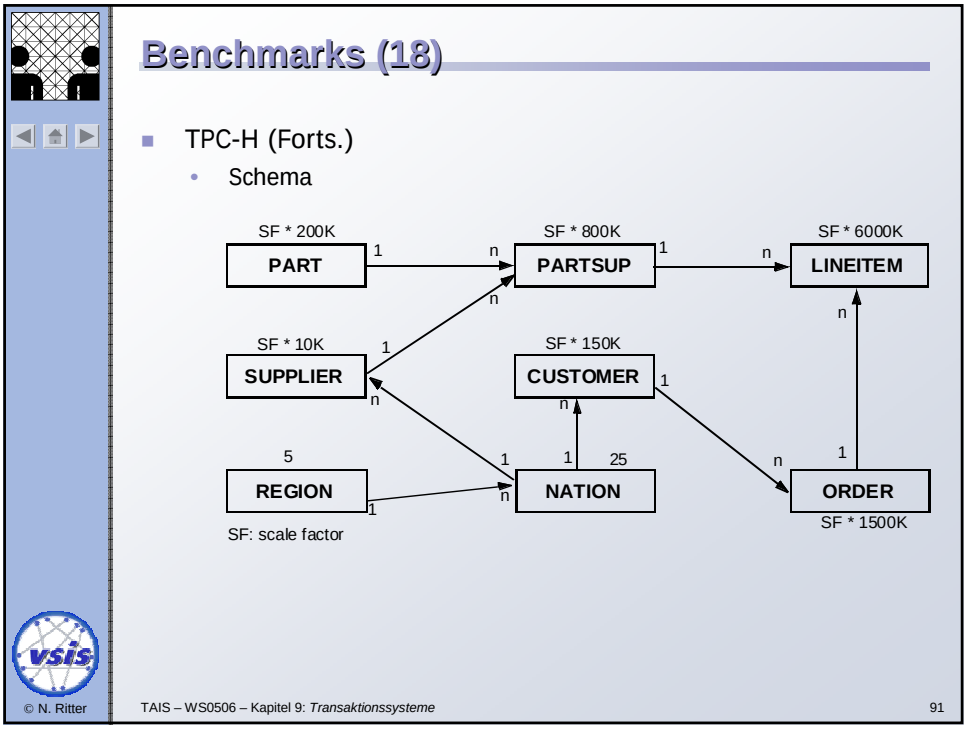
- TPC-H und TPC-R
  - Benchmarks zur Auswertung komplexer Anfragen in großen DBs
    - ursprünglich als TPC-D eingeführt (gültig bis 4/99)
    - bilden durch Anfragen typische Aktivitäten im Großhandel nach
    - zielen auf die Datenanalyse ab
      - Berechnung von Trends
      - Unterstützung der Entscheidungsfindung
    - übernehmen Schema, Skalierungsfaktoren und Anfragen vom TPC-D
    - erweitern sie auf 22 Anfragetypen und 2 Aktualisierungsfunktionen (DB refresh)
  - TPC-H (ad-Hoc, decision support) nutzt kein Vorwissen aus (lange Ausführungszeiten für Anfragen)
  - TPC-R (business Reporting, decision support) unterscheidet sich vom TPC-H durch Ausnutzung von Vorwissen: DBS kann speziell auf Standard-Anfragen hin optimiert werden!



Benchmarks (17)

- TPC-H und TPC-R (Forts.)
  - Einsatzumgebung



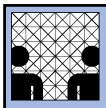


Benchmarks (18)

- TPC-H (Forts.)
  - Schema

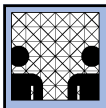
Benchmarks (19)

- TPC-H (Forts.)
  - Anfragebeispiel Q9: Product Type Profit Measure Query
    - The query finds, for each nation and each year, the profit for all parts ordered in that year which contain a specified substring in their names and which were filled by a supplier in that nation. The profit is defined as the sum of
      (L_EXTENDEDPRICE*(1-L_DISCOUNT)) –
      (PS_SUPPLYCOST * L_QUANTITY)]
      for all lineitems describing parts in the specified line. The query lists the nations in ascending alphabetical order and, for each nation, the year and profit in descending order by year (most recent first).



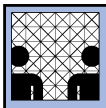
Benchmarks (20)

- TPC-H (Forts.)
  - Verfügbarkeit der Datenbank: 24*7*52 h (mit Wartungspausen)
  - Maßeinheiten:
    - Composite Query-per-Hour Metric:
      QphH@Size = sqrt(Power@Size * Throughput@Size)
      - query processing power at the chosen DB size (Power@Size) within a single query stream
      - throughput at the chosen DB size (Throughput@Size) in multi-user environment
    - Price/Performance Metric:
      Price-per-QphH@Size = $/QphH@Size



Benchmarks (21)

- TPC-H (Forts.)
  - Anfrage Q2: Minimum Cost Supplier Query
    - The query finds, in a given region, for each part of a certain type and size, the supplier who can supply it at minimum cost. If several suppliers in that region offer the desired part type and size at the same (minimum) cost, the query lists the parts from suppliers with the 100 highest account balances. For each supplier, the query lists the supplier's account balance, name and nation, the part's number and manufacturer, the supplier's address, phone number and comment information.

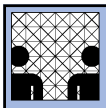


Benchmarks (22)

- TPC-H (Forts.)
  - Anfrage Q2: SQL-Umsetzung

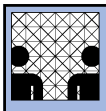
Note: return only the first 100 selected rows

```
SELECT  S_ACCTBAL, S_NAME, N_NAME, P_PARTKEY, P_MFGR, S_ADDRESS,
        S_PHONE, S_COMMENT
FROM    PART, SUPPLIER, PARTSUPP, NATION, REGION
WHERE   P_PARTKEY = PS_PARTKEY
  AND S_SUPPKEY = PS_SUPPKEY
  AND P_SIZE = [size]
  AND P_TYPE LIKE '%[type]'
  AND S_NATIONKEY = N_NATIONKEY
  AND N_REGIONKEY = R_REGIONKEY
  AND R_NAME = '[region]'
  AND PS_SUPPLYCOST =
  (SELECT MIN(PS_SUPPLYCOST)
          FROM    PARTSUPP, SUPPLIER, NATION, REGION
          WHERE   P_PARTKEY = PS_PARTKEY
            AND S_SUPPKEY = PS_SUPPKEY
            AND S_NATIONKEY = N_NATIONKEY
            AND N_REGIONKEY = R_REGIONKEY
            AND R_NAME = '[region]')
ORDER BY S_ACCTBAL DESC, N_NAME, S_NAME, P_PARTKEY;
```



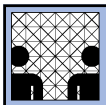
Benchmarks (23)

- TPC-W
  - Benchmark für *Web Commerce*
    - Modellierung/Abwicklung von typischen Aktivitäten des E-Commerce wie Browsing, Bestellung und Bezahlung von Waren, ...
    - Vermarktung und Verkauf von Produkten oder Dienstleistungen übers Internet (Einzelhandel, Flug-/Reisereservierung usw.)
    - Repräsentation von Web-Anwendungsumgebungen mit
      - ACID-Eigenschaften
      - geschützter Interaktion
      - Konsistenz und dynamischer Generierung von Web-Seiten
    - Netzverkehr und Benutzerinteraktion bleiben (wegen der stark schwankenden und nicht kontrollierbaren Web-Lasten) bei der Leistungsmessung ausgeklammert
  - TPC-W kann als Server-Benchmark aufgefasst werden (Back-End-Messung)



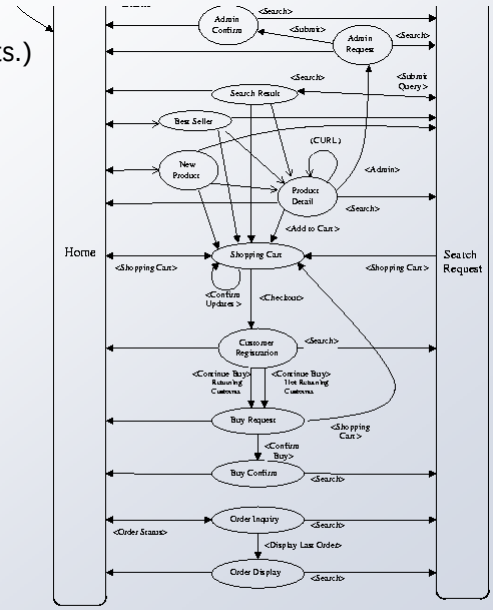
Benchmarks (24)

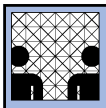
- TPC-W (Forts.)
  - Eigenschaften
    - Spezifikation von V1.1 ca. 200 Seiten
    - komplexe Definition von Datenstrukturen, Skalierungsvorschriften, … (www.tpc.org/wspec.html)
    - Überblick über Web-Interaktionen
  - Leistungsmaße
    - WIPS (Web Interactions Per Second) bezieht sich auf ein durchschnittliches Shopping-Szenario
    - $/WIPS
      - Systemkosten umfassen nur die Komponenten auf der Server-Seite: Web-, Anwendungs- und DB-Server sowie die erforderlichen Kommunikationskomponenten



Benchmarks (25)

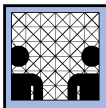
- TPC-W (Forts.)





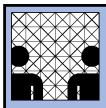
Benchmarks (26)

- TPC-Benchmarks: Kennzahlen (http://www.tpc.org)
  - Viele Benchmark-Messungen durch 33 Hersteller
    - TPC-A (TPC-B): >300 (>130) publizierte Ergebnisse von 115 (73) verschiedenen Systemen
    - zwei “goldene Zahlen”: tps und $/tps
    - TPC-A und TPC-B wurden ab 6/1995 nicht mehr verwendet
  - Leistungswerte für TPC-A
    - ca. 100 - 200 KInstr. / TA (anfangs bis zu 1 Mill. Instr. pro TA)
    - 2 E/A-Vorgänge pro TA (anfangs bis zu 20)
    - 1990: 33 tpsA zu 25,500 $/tpsA
    - 1995: 3692 tpsA zu 4,873 $/tpsA
    - 111 und 5 als Verbesserungsfaktoren



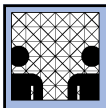
Benchmarks (27)

- TPC-Benchmarks: Kennzahlen (Forts.)
  - Leistungswerte für TPC-B
    - ca. 75 KInstr. / TA
    - 1991: 103 tpsB zu 4,167 $/tpsB
    - 1994: 2,025 tpsB zu 254 $/tpsB
    - 19 und 16 als Verbesserungsfaktoren
  - Warum hatten diese TPC-Benchmarks solche Erfolge?
    - erste Benchmark-Messungen ohne spezielle Optimierung
    - reale Performance-Steigerungen durch HW- und SW-Produkte
    - Systemverbesserungen, um durch Benchmark aufgedeckte Performance-Schwächen zu eliminieren
    - effektive Nutzung des “Benchmark-Games”: Hersteller lernten voneinander, wie der Benchmark am besten abzuwickeln ist



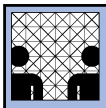
Benchmarks (28)

- TPC-Benchmarks: Kennzahlen (Forts.)
  - Leistungswerte für TPC-C
    - 1992: 54 tpmC zu 188,562 $/tpmC
    - 1998: 52,871 tpmC zu 135 $/tpmC
    - 2000: 505,302 tpmC zu 21 $/tpmC
    - 2001: 709,220 tpmC zu 14.96 $/tpmC (TPC-C Version 5)
    - 13,134 und 12,604 als Verbesserungsfaktoren
  - Leistungswerte für TPC-C (Version 5) bei Price/Performance
    - 2002: 16,756 tpmC zu 2.78 $/tpmC
    - 2003: 82,226 tpmC zu 2.76 $/tpmC



Benchmarks (29)

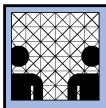
- TPC-Benchmarks: Kennzahlen (Forts.)
  - Leistungswerte für TPC-D (bei 100 GB)
    - 1995: 84 QphD und 52,170 $/QphD
    - 1998: 1,205 QphD und 1,877 $/QphD
    - 14 und 28 als Verbesserungsfaktoren (bis 1998)
    - ab 1999: TPC-H und TPC-R;
      sie sind gegenüber TPC-D von 17 auf 22 Anfragen erweitert
  - Leistungswerte für TPC-R
    - 2000: 21,254 QphR und 607 $/QphR bei 1000GB
    - 2003: 4,442 QphR und 35 $/QphR bei 100 GB



Benchmarks (30)

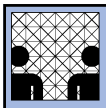
- TPC-Benchmarks: Kennzahlen (Forts.)
  - Leistungswerte für TPC-H
    - 2000: 1,699 QphH und 161 $/QphH bei 100 GB
    - 2002: 5,578 QphH und 358 $/QphH bei 100 GB
    - 2000: 12,866 QphH und 670 $/QphH bei 1,000 GB
    - 2002: 25,805 QphH und 203 $/QphH bei 1,000 GB
    - 2002: 27,094 QphH und 240 $/QphH bei 3,000 GB (V1)
    - 2002: 81,501 QphH und 243 $/QphH bei 10,000 GB (V2)

Note: The TPC believes that comparisons of TPC-H/R results measured against different database sizes are misleading and discourages such comparisons. The TPC-H/R results shown below are grouped by database size to emphasize that only results within each group are comparable.



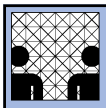
Benchmarks (31)

- TPC-Benchmarks: Kennzahlen (Forts.)
  - Leistungswerte für TPC-W
    - 2000: 1,262 WIPS und 277 $/WIPS
    - 2002: 21,139 WIPS und 32.62 $/WIPS (Item Count 10,000)
    - 2002: 10,439 WIPS und 106.73 $/WIPS (Item Count 100,000)
  - Bemerkung
    - Bei „Transaktionen“ (besser *mouse clicks*) übers Internet wurde neuerdings scherzhaft das Kostenmaß „m$/tps“ geprägt.



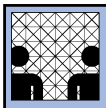
Zusammenfassung (1)

- Allgemeine Aspekte des Schichtenmodells
  - Schichtenmodell ist allgemeines Erklärungsmodell für die DBS-Realisierung
  - Schichtenbildung lässt sich zweckorientiert verfeinern/vergröbern: Anwendbarkeit für TA-Systeme, Verteilte TA-Systeme, DBS, ...
  - Entwurf geeigneter Schnittstellen erfordert große Implementierungserfahrung
  - Konkrete Implementierungen verletzen manchmal die Isolation der Schichtenbildung aus Leistungsgründen (→ kritische Funktionen haben „Durchgriff“)
- Klassifikation Verteilter TA-Systeme
  - Transaction Routing, Programmierte Verteilung, Aufruf von DB-Operationen
  - Mehrrechner-DBS als Shared-Nothing oder Shared-Disk-Systeme
  - Aber: Mehrrechner-DBS verlangen heute in der Regel ein (geeignetes) Datenmodell und sind nicht überall verfügbar
  - Wie geht man vor, um heterogene Datenbanksysteme in die Anwendungssysteme zu integrieren?



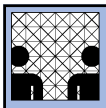
Zusammenfassung (2)

- Entwicklung verteilter Anwendungen: vor allem Programmierte Verteilung
  - TP-Monitor übernimmt Kommunikation und Transaktionsverwaltung
  - Unterstützung von Knotenautonomie und Heterogenität
  - keine Verteilungstransparenz
  - eingeschränkte Flexibilität des Datenzugriffs



Zusammenfassung (3)

- X/OPEN DTP
  - Interoperabilität über standardisierte Schnittstellen und Protokolle
  - für die lokale Zusammenarbeit von Anwendung, TA-Mgr und RMs
  - und für die verteilte TA-Verarbeitung.
- TP-Monitore
  - liefern den „Klebstoff“ in komplexen C/S-Umgebungen
  - erlauben die Umsetzung des Transaktionskonzeptes als Grundlage
    zur Realisierung zuverlässiger verteilter Systeme
  - sind Spezialisten für Prozess-, Transaktions- und C/S-Kommunikations-Verwaltung



Zusammenfassung (4)

- TA-Systeme werden „überall“ eingesetzt
  - Sie stellen eine ausgereifte Technologie dar
  - OLTP-Hochleistungssysteme sind in vielen Varianten (Plattform, TP-Monitor, Datenbanksystem) auf dem Markt verfügbar
  - Es ist ein Markt von > 10^{11} $/Jahr
- Es gibt eine zunehmende Vielfalt an TPC-Benchmarks
  - Standardisierung von Arbeitslasten für TA-Systeme
  - aktueller Benchmark für komplexe OLTP-Anwendungen: TPC-C
  - aktueller Benchmark für *Decision Support* und *ad hoc:* TPC-H
  - TPC-W als Benchmark für Web-DB-Anwendungen ‚wird gerade abgelöst‘ durch TPC-App (Applikationsserver)
  - „dramatische“ Verminderung der $/tps-Kosten
  - Der Wettbewerb hat funktioniert!