

Chapter 4

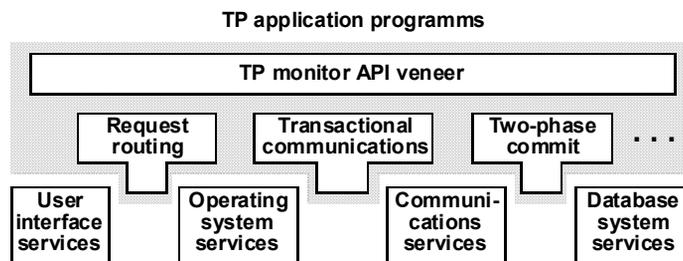
TP-Monitors and Distributed Transactions



Middleware for Heterogeneous and Distributed Information Systems - WS04/05

Aufgaben des TP-Monitors

- Bewältigung der Heterogenität
- Kontrolle der Kommunikation
- Verwaltung der Terminals
- Präsentationsdienste
- Kontextverwaltung
- Start/Restart
- Programmverwaltung
- Konfigurationsverwaltung
- Lastbalancierung
- Autorisierung
- Bereitstellung von Administrationsschnittstellen

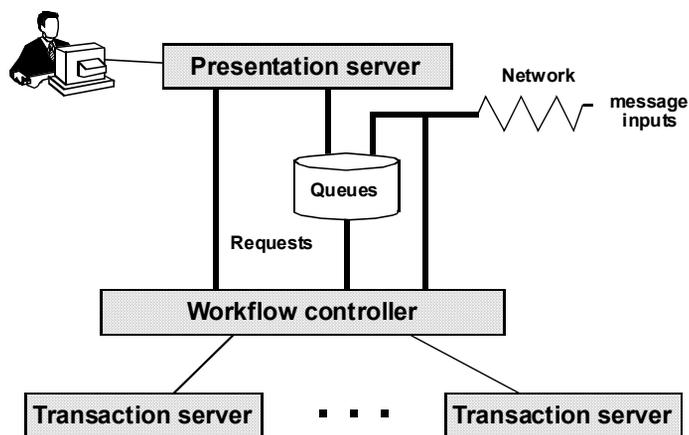


TP-Monitore - Anforderungen

- Anforderungen (an die Realisierung eines TP-Monitors)
 - große Anzahl angeschlossener Terminals ($10^2 - 10^4$)
 - konkurrierende Ausführung vieler Funktionen
 - Zugriff auf gemeinsame Daten mit
 - größtmöglicher Aktualität
 - Erhaltung der Konsistenz
 - Zugriffskontrolle
 - hohe Verfügbarkeit
 - kurze Antwortzeiten
 - Ausfallsicherheit
 - flexible Reaktion auf stochastisches Verkehrsaufkommen
 - Unterstützung der Administration bei:
 - Systeminstallation
 - Änderungen und Ergänzungen
 - Leistungsüberwachung und Tuning



Logische Architektur (vereinfachte Sicht)

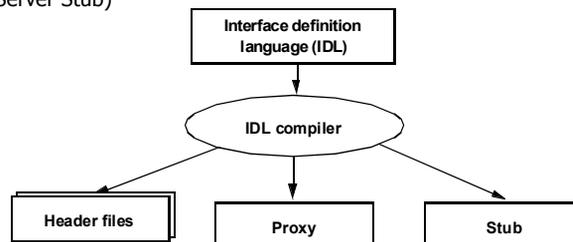


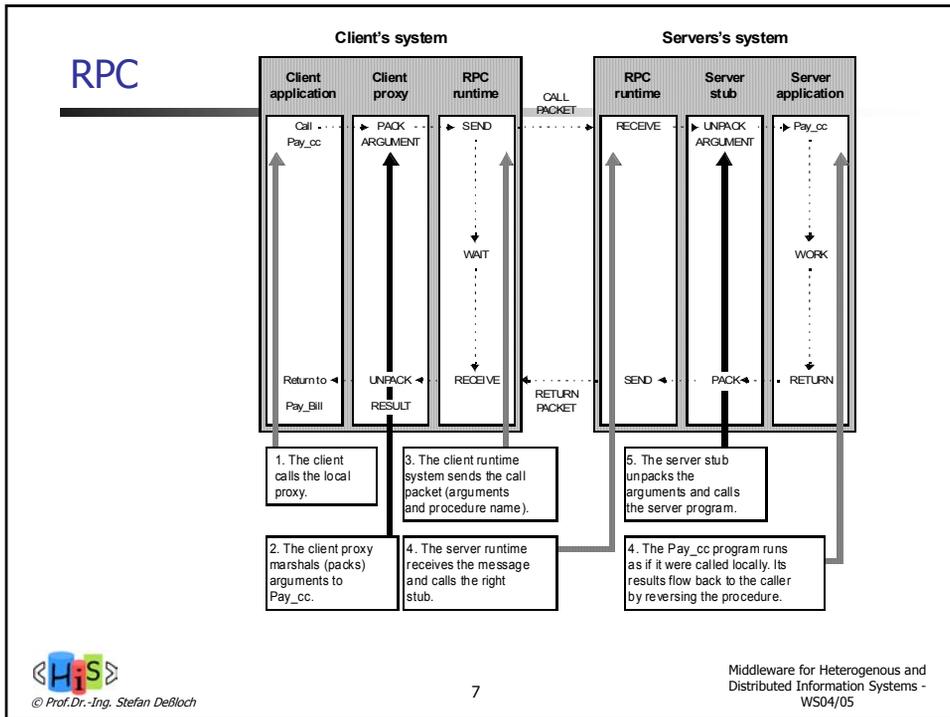
Kommunikation und Verteilung

- Ein verteiltes (Informations-)System
 - besteht aus autonomen Subsystemen, die koordiniert zusammenarbeiten, um eine gemeinsame Aufgabe zu erfüllen
- Kommunikation zwischen Subsystemen:
 - Remote Procedure Calls (RPC)
 - Peer-To-Peer-Messaging
 - Message Queuing
- Garantie der ACID-Eigenschaften für verteilte Abläufe
 - Server/Systemkomponenten sind RMs
 - (Transaction) Remote Procedure Calls (TRPC)
 - Verteilte Transaktionsverwaltung

Remote Procedure Call (RPC)

- Ziel: Einfaches Programmiermodell für verteilte Applikationen
 - basiert auf Prozedur als Aufrufmechanismus für vert. Komponente
- Zentraler Mechanismus in fast allen Formen von Middleware
- Wichtige Konzepte
 - Interface Definition Language
 - Proxy (Client Stub)
 - Stub (Server Stub)



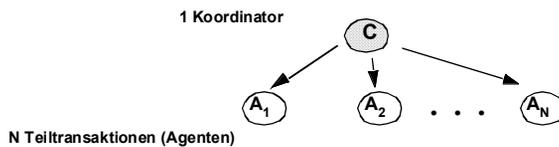


Transaktionaler RPC (TRPC)

- Server sind RMs
- RPCs erfolgen im Kontext einer Transaktion
 - Demarkation (BOT, EOT) i.a. durch Client
- TRPC-Stub
 - wie RPC-Stub
 - zusätzlich jedoch als **Agent des TP-Monitors** verantwortlich für die TA-orientierte Kommunikation
- TRPC erfordert folgende TP-Monitor-Aktivitäten
 - Binden von RPC an Transaktionen durch ihre Kennzeichnung mit TRID
 - Informieren des TA-Mgr über den RM-Aufruf, wenn der TP-Monitor diesen verschickt (Registrieren der Teilnehmer an einer Transaktion)
 - Binden der Prozesse an Transaktionen: im Fehlerfall (Crash) kann der TP-Monitor den TA-Mgr über den Ausfall des Prozesses informieren

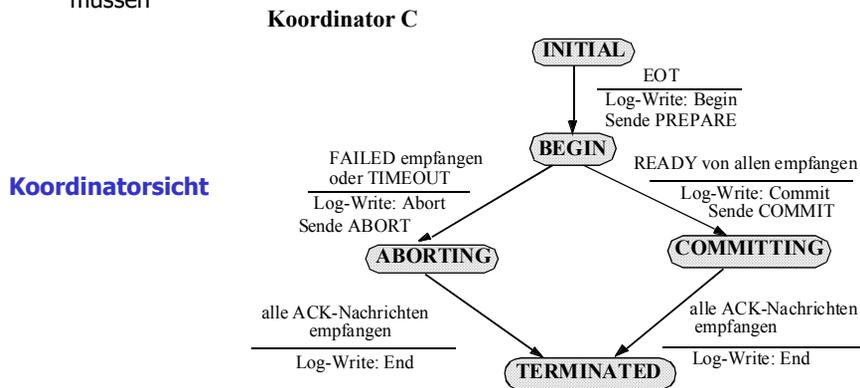
Verteilte Transaktionen

- Erfordern rechnerübergreifendes (mehrphasiges) Commit-Protokoll
 - Sicherstellen der Atomizität einer globalen Transaktion
 - Anforderungen an geeignetes Commit-Protokoll:
 - geringer Aufwand (#Nachrichten, #Log-Ausgaben)
 - minimale Antwortzeitverlängerung (Nutzung von Parallelität)
 - Robustheit gegenüber Rechnerausfällen und Kommunikationsfehlern
 - Erwartete Fehlersituationen
 - i. allg. partielle Fehler (z.B. Verbindung fällt aus, ...)
 - Transaktionsfehler
 - Systemfehler (Crash)
 - Gerätefehler
 - Fehlererkennung z. B. über Timeout



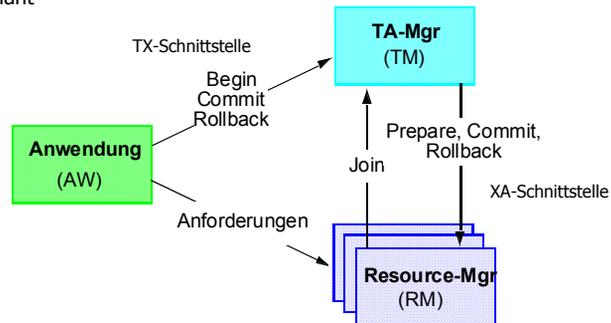
Zweiphasen-Commit

- Prepare-Phase, Commit/Abort-Phase
- erfordert Folge von Zustandsübergängen, die sicher (Log) vermerkt werden müssen



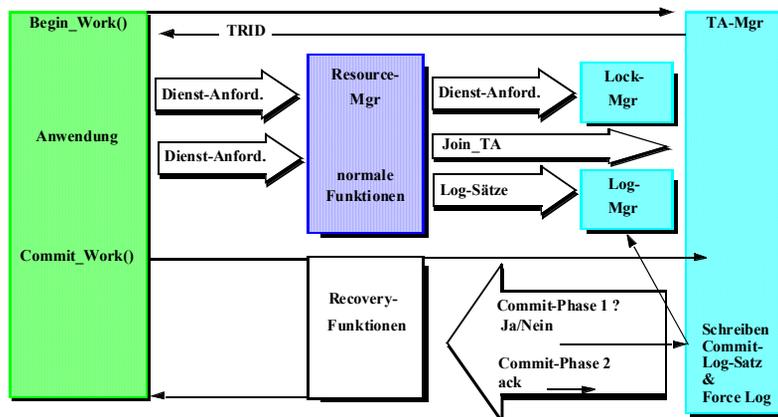
Verteilte Transaktionsverwaltung nach X/OPEN DTP

- unabhängige TA-Mgr
- Resource Manager
 - recoverable
 - XA-compliant

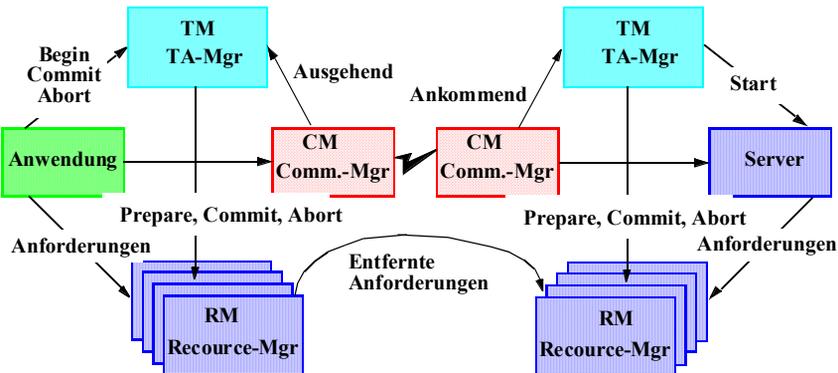


Zentrale Transaktionsdienste

- Rolle der zentralen Transaktionsdienste



Verteilte Transaktionsverwaltung nach X/OPEN DTP (2)

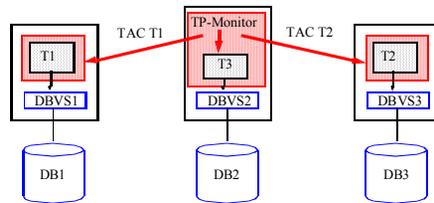


Verteilung unter Kontrolle des TP-Monitors

- TP-Monitor verbirgt weitgehend Heterogenität bezüglich Kommunikationsprotokollen, Netzwerken, BS und Hardware
- DBS bleiben weitgehend unabhängig (keine Kooperation zwischen DBS)
- heterogene DBS möglich
- geringe Implementierungskomplexität
- 3 wesentliche Alternativen:
 - Transaction Routing
 - Einheit der Verteilung ist die Transaktion
 - Programmierte Verteilung
 - Einheit der Verteilung ist die Teil-Transaktion
 - Verteilung von DB-Operationen
 - Einheit der Verteilung ist die DB-Anfrage

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
- keine Kooperation zwischen DBVS
 - pro Rechner eigene DB/Schemata
 - heterogene DBS möglich
- als alleiniges Verteilgranulat zu inflexibel
 - keine echt verteilte Transaktionsausführung

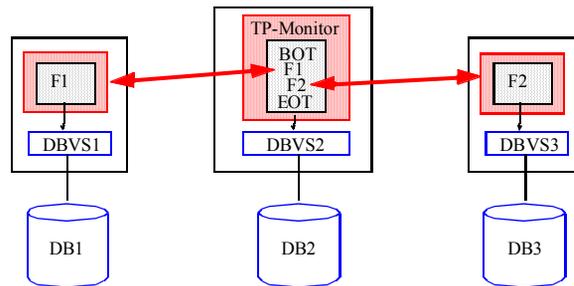


Programmierte Verteilung

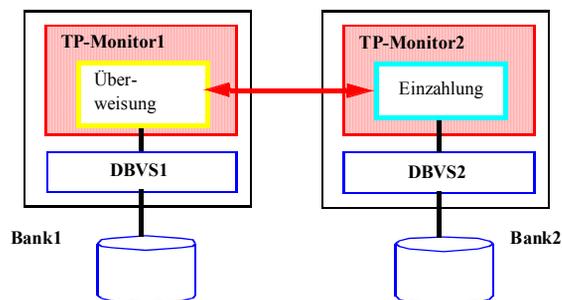
- Verteilung auf Ebene von Anwendungsprogrammen
 - Zugriff auf externe DB durch Aufruf eines Teilprogramms (RPC) auf dem betreffenden Rechner
 - jedes Teilprogramm greift nur auf lokale DB zu (mit jeweiliger Anfragesprache)
- Transaktionsverwaltung
 - TP-Monitor koordiniert verteiltes Commit-Protokoll
 - 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



Programmierte Verteilung (2)



Programmierte Verteilung (3)



Programmierte Verteilung (4)

```
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
```

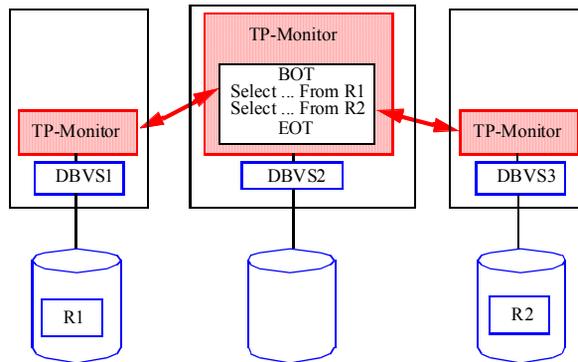


Verteilung von DB-Operationen

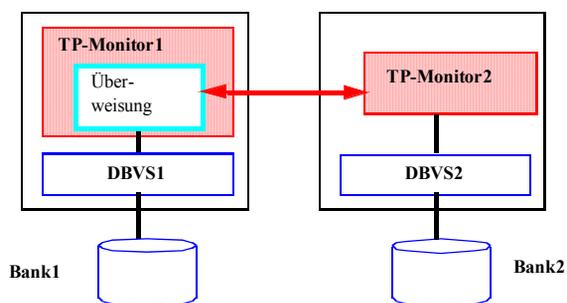
- DB-Operationen als Verteileinheiten
 - AP können auf entfernte Datenbanken zugreifen
 - Programmierer sieht mehrere Schemata; jede Operation muß innerhalb eines Schemas (DB-Partition) ausführbar sein
 - gemeinsame Anfragesprache wünschenswert
 - Ort der Verteilung kann für AP prinzipiell transparent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



Verteilung von DB-Operationen (Forts.)



Verteilung von DB-Operationen (Forts.)



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
```



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)
 - keine rechnerübergreifenden DB-Operationen
 - geringes Maß an Verteilungstransparenz



Bewertung (2)

- 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
 - DB-Gateways



Zusammenspiel der Transaktionsdienste

- strikte Unterscheidung zwischen **TP-Monitor** und **TA-Mgr**
 - viele Systembenutzer benötigen den TP-Monitor nicht: Stapel-AW, Ad-hoc-Anfragen über eine direkte SQL-Schnittstelle
 - spezielle AW-Systeme (z. B. CAD) haben ihre eigene Terminalumgebung
 - alle Aktivitäten brauchen jedoch Transaktionsunterstützung
- eine Konsequenz ist die Trennung der Komponenten zur Durchführung
 - der Transaktionskontrolle (TA-Mgr) und
 - des transaktionsorientierten Betriebsmittel-Scheduling (TP-Monitor)



Zusammenfassung

- Middleware-Aspekte
 - TA-Konzept als Basis für (DB-)Middleware
 - TA-Schutz für verteilte Abläufe notwendig
 - damit sind die Prinzipien der Verteilten Transaktionsverwaltung grundlegend für jede Art der Middleware-Integration
 - TA-Manager/TP-Monitor als notwendige Komponente jeder DB-Middleware
 - TP-Monitor selbst gehört zur DB-Middleware
 - Integration von verteilten, heterogenen RM
 - Herstellung von globalen Sichten (TA-Routing, Vert. Progr., Vert. v. DB-Ops.)
 - jedoch kein integriertes Schema und keine verteilte Bearbeitung einzelner DB-Operationen
- Der TP-Monitor ist ein RM,
 - der andere RM und Ressourcen wie Prozesse, Tasks, Zugriffsrechte, Programme und Kontexte verwaltet