

Kapitel 9 Message-oriented Middleware (MOM)

Inhalt:

Warteschlangen in TP-Monitoren
Message Queuing Systeme
Stratifizierte Transaktionen
Message Brokering
Zusammenfassung



Middleware for Heterogenous and Distributed Information Systems - WS04/05

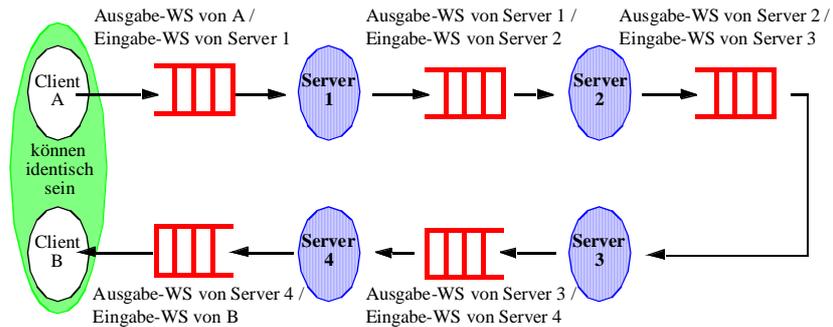
Einsatz von Warteschlangen in TP-Monitoren

- Lastkontrolle
 - Normalisierung von Lastspitzen
 - statt Erzeugung eines weiteren Prozesses wird Auftrag in (flüchtiger) Schlange (vor der Server-Klasse) vermerkt
- Endbenutzerkontrolle
 - Aushändigung der Ausgaben von asynchronen Vorgängen ist kritisch (Text, Bilder, Geld, ...)
 - Wiederholung der Ausgabe kann erforderlich werden, solange der Empfang nicht explizit quittiert wurde
- Wiederherstellbare Dateneingabe
 - Anwendungen, die durch Eingabenachrichten hoher Frequenz getrieben werden und für hohen Durchsatz ausgelegt sind
 - Eingabe wird von einer Warteschlange gelesen
 - Eingabe darf nicht verloren gehen; auch nicht bei Crash
- Multitransaktionale Anforderungen



Multitransaktionale Anforderungen

- aufeinanderfolgende Transaktionen, die auf hohen Durchsatz ausgelegt sind
 - Voraussetzung: keine zusätzlichen Interaktionen mit Endbenutzer erforderlich
- basierend auf wiederherstellbaren Eingabedaten



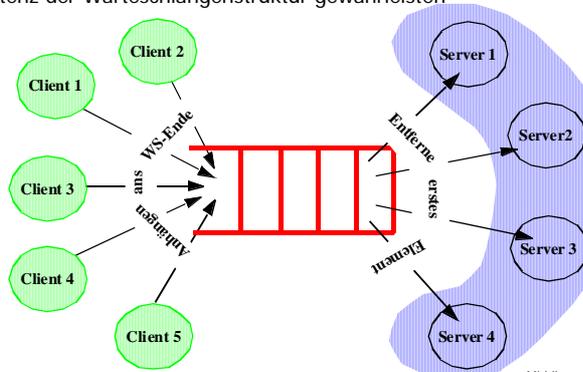
© Prof. Dr.-Ing. Stefan Deßloch

3

Middleware for Heterogenous and Distributed Information Systems - WS04/05

Flüchtige Warteschlangen

- Behandlung von Überlast
 - Client-seitiges Modell: direkte, synchrone Kommunikation
 - Anforderungen werden vom TP-Monitor kurzzeitig in serverklassenspezifischer WS abgelegt
 - es muss "exactly-once" zugesichert werden; konkurrierende Zugriffe müssen Konsistenz der Warteschlangenstruktur gewährleisten



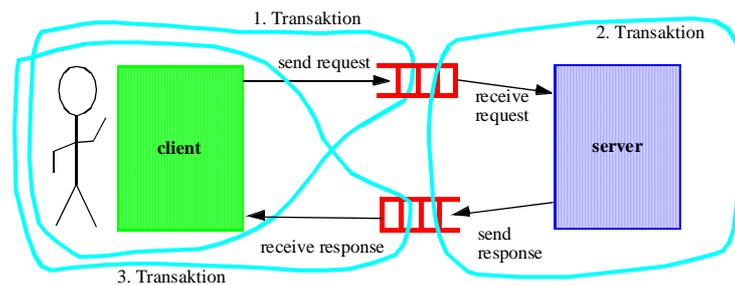
© Prof. Dr.-Ing. Stefan Deßloch

4

Middleware for Heterogenous and Distributed Information Systems - WS04/05

Asynchrone Transaktionsverarbeitung

- Entkopplung von Eingabe, Verarbeitung und Ausgabe
 - Einsatz von drei oder mehr separaten TA
 - Durchsatzoptimierung anstelle von Antwortzeitminimierung
 - Anforderungs- und Ergebnis-WS sind **dauerhafte** Objekte
 - Ein- und Auslagern muß jeweils Teil der zugehörigen TA sein
 - Anforderung wird erst bei Commit der einlagernden TA für andere TA sichtbar
 - Beim Scheitern der auslagernden TA wird Anforderung wieder auf die WS "zurückgelegt"



Message Queuing Systems (MQS)

- Haben sich aus Queuing-Systemen der TP-Monitore entwickelt
- Botschaftenorientierte Interoperabilität
 - Programmiermodell: Kommunikation durch Austausch von Botschaften
- Bereitstellung persistenter Warteschlangen
 - zur Abwicklung asynchroner Kommunikation
 - als zuverlässige Nachrichtenpuffer
- Lose Kopplung von Anwendungskomponenten
 - "Client" ist während der Abarbeitung einer Anforderung nicht blockiert
 - "Server" kann Zeitpunkt der Anforderungsbearbeitung flexibel wählen
 - ist mglw. zum Zeitpunkt der Anforderungsabgabe garnicht verfügbar
- Transaktionale MQS ("reliable MQ")
 - persistente Warteschlangen
 - garantierte "exactly-once"-Semantik
 - Einlagern bzw. Auslagern von Botschaften im Kontext von Transaktionen

Interaktion mit MQS

- Grundlegende Operationen:
 - Enqueue: Einlagern von Nachrichten in eine Warteschlange
 - i.A. nicht blockierend
 - Dequeue: Auslagern von Nachrichten
 - meistens blockierend
- Variationen
 - Shared Queues
 - mehrere Applikationen können auf die gleiche MQ zugreifen
 - Bsp.: Lastbalancierung durch Einsatz von mehreren "Serverapplikationen"
 - Zusätzliche Eigenschaften für Botschaften
 - Priorität, Time-outs, ...

JMS – Standardisierte Interaktion mit MQS

- Java Messaging Service (JMS)
 - Standard-API für Message Queuing, Message Brokering
- Funktionsumfang (für Message Queuing)
 - Aufbau von Verbindung zum Queuing Service
 - Send, Receive (enqueue/dequeue)
 - selektives receive möglich
 - Transaktionales MQ
 - Erhaltung der Botschaftenreihenfolge im Rahmen einer "Session"
- Aufbau einer JMS Message
 - Header (standard)
 - message-id, correlation-id, delivery mode (persistent/not persistent), destination (queue), priority, redelivered, reply-to, timestamp
 - Properties (optional)
 - anwendungsspezifische, herstellerspezifische und optionale standardisierte Eigenschaften
 - Body
 - eigentlicher Inhalt
 - versch. Inhaltstypen (Bytes, Text, Java Objekt, ...)

Stratifizierte Transaktionen (1)

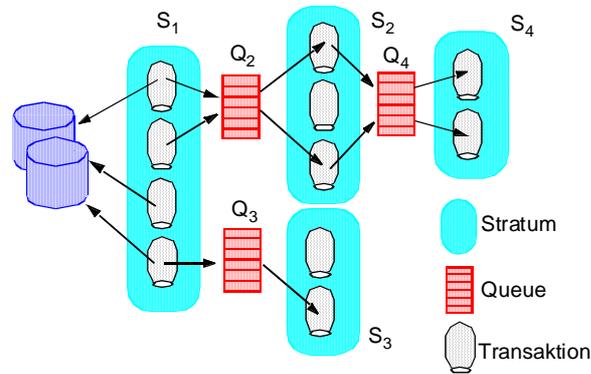
- AW-orientierte Zerlegung der Transaktion T
 - in T_1, \dots, T_n ;
 - Verkettung: jede T_i erhält persistente Warteschlange Q_i , aus der sie Anforderungen erhält, bestimmte Operationen auszuführen
 - lineare Reihenfolge nicht zwingend
- WICHTIG:
 - alle von den Transaktionen T_i manipulierten Ressourcen (also insbes. auch die Nachrichten) sind wiederherstellbar
 - dies bedeutet, daß sich die von den Transaktionen T_i benutzten Resource-Manager (DBVS, MOM) in atomares Commit einbinden lassen (XA-Protokoll, 2PC)

Stratifizierte Transaktionen (2)

- Struktur stratifizierter Transaktionen
 - einige T_i sollen gemeinsam zum erfolgreichen Ende kommen
 - disjunkte, vollständige Zerlegung von T in Transaktionsmengen S_1, \dots, S_m
 - $S_i \subseteq T$ mit $S_i \neq \emptyset$ und $S_i \cap S_j = \emptyset$ für $i \neq j$ und $\bigcup_{j=1}^m S_j = T$
 - Transaktionen von S_i werden durch 2PC-Protokoll synchronisiert
 - Menge S_i von Transaktionen heißt Stratum

Stratifizierte Transaktionen (3)

- Verkettung der Strata innerhalb der stratifizierten Transaktion T durch Baumstruktur



Stratifizierte Transaktionen (4)

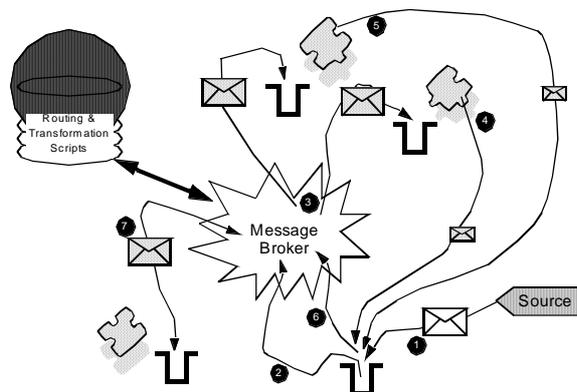
- Verhalten
 - alle Strata führen schließlich Commit aus unter der Bedingung, dass die jeweiligen Vater-Strata vorher Commit ausgeführt haben
 - falls Stratum wiederholt scheitert (echte Ausnahme): stratifizierte Transaktion muss zurückgesetzt werden (Kompensation)
- Vorteile:
 - im Vergleich zu T: früheres Commit der einzelnen Strata S_i; dies impliziert frühere Freigabe der Sperren und damit höhere Nebenläufigkeit
 - Antwortzeit für Benutzer: Ausführungszeit des Wurzelstratums
 - 2PC-Protokolle für alle Strata können weniger Nachrichten umfassen als das 2PC-Protokoll einer globalen Transaktion (Kokolation als mögliches Kriterium für Stratifikation von T: lokale 2PC-Nachrichten)

Message Brokering (1)

- bisher (Message Queuing):
 - explizite Zieldefinition (point-to-point)
 - vereinbarte Nachrichtenstruktur
- nun (Message Brokering):
 - Verteilung von Nachrichten ohne Zieldefinition (hub-and-spoke)
 - identische Struktur von gesendeter und empfangener Nachricht wird nicht vorausgesetzt
 - Publish/Subscribe
 - Regeln zur Weiterleitung von Nachrichten ausgehend von ihrem Inhalt
 - Anpassung von Format und Inhalt an 'Empfängerwünsche'

Message Brokering (2)

- Abläufe



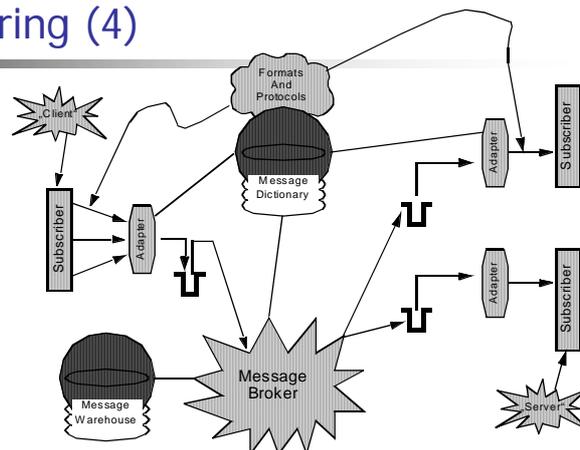
Message Brokering (3)

- Charakteristika
 - Quellen: Publishers
 - Senken: Subscribers
 - nach 'Einreichung' einer Nachricht bestimmt der Broker die Empfänger aufgrund der 'Subscribe-Spezifikationen'
 - Subscription:
 - repräsentiert Interesse eines Empfängers an einer Nachricht ausgehend von ihrem Originalinhalt
 - definiert Format und Inhalt der 'Zustellung'
 - inhaltliche Abweichungen
 - Annotierung
 - 'Reinigung'
 - Filter
 - möglicherweise komplizierte Nachrichtentransformationen erforderlich



Message Brokering (4)

- Systemstruktur



- Dictionary: Formatspezifikationen, Transformationsregeln/-funktionen/-Skripte
- Warehouse: weitere Auswertung und Analyse von Nachrichten

*F. Leymann: A Practitioners Approach to Data Federation
F. Leymann, D. Roller: Production Workflow, Prentice Hall, 2000.*



Zusammenfassung (1)

- Integration durch MOM
- Message Queuing
 - Nutzung
 - TP-Monitore, z. B. QTP in IMS/DC
 - WfMS, z. B. MQSeries-Workflow (IBM)
 - eigenständige Middleware zur asynchr. Kommunikation
 - DB-Aspekt
 - Persistenz der Nachrichten
 - Einbindung von Queue-Operationen in Sender- und Empfängertransaktionen
 - Folgen von Queue-Operationen als semantische Einheiten

Zusammenfassung (2)

- Message Queuing (Forts.)
 - Funktionalität
 - allgemeine Dienste
 - Enqueue, Dequeue, Browsing, ...
 - verschiedene Message Types
 - anwendungsspezifische Prioritäten
 - inhalts-basierte Selektion
 - einstellbare Transaktionalitäts- und Persistenzeigenschaften
 - Message Brokering
 - Publish/Subscribe
 - Nachrichtentransformation