JAVA SPACES

Beschreibe das Transaktionsmodell von Java Spaces

ACID Eigenschaften:
- Atomicity
- Consistency
- Isolation
- Durability

Pessimistische Concurrecy Control
- Operationen setzen Locks (Read, Write und Delete Locks)

Isolation Level
- Repeatable Read
  - gelesene Daten werden gelockt und nur committete Änderungen sind für andere Transaktionen sichtbar
- Phantom Reads möglich!
  - Wiederholte Lesevorgänge können unterschiedliche Ergebnisse liefern

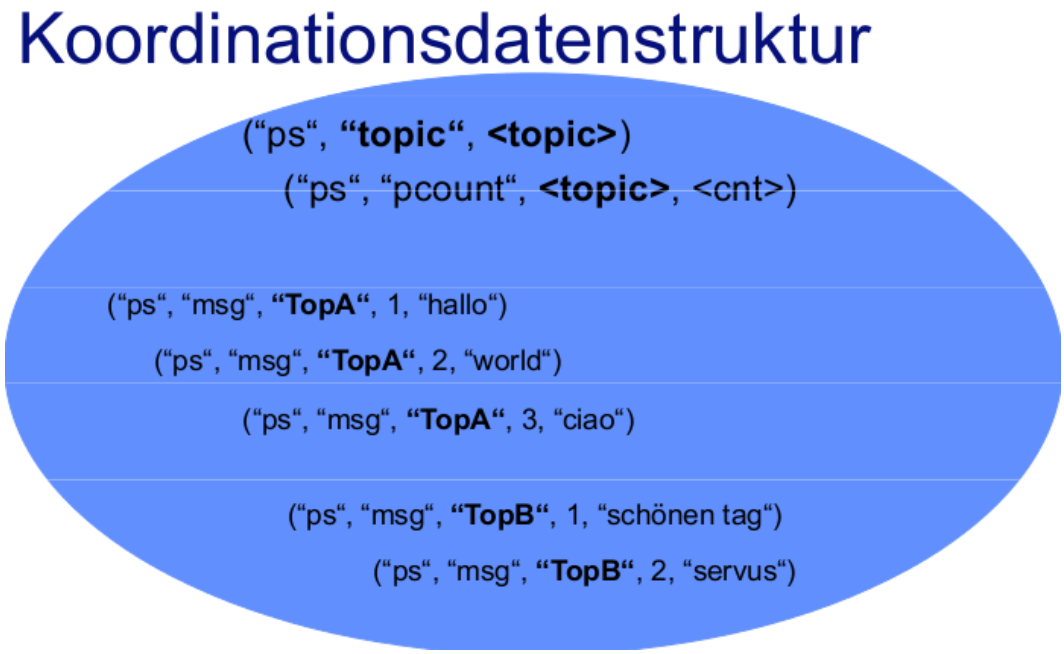
Sind 2 gleiche Resultate bei read möglich + Erklärung

Ja, aufgrund des Isolation Levels Repeatable Read: gelesene Daten werden gelockt und nur committete Änderungen sind für andere Transaktionen sichtbar

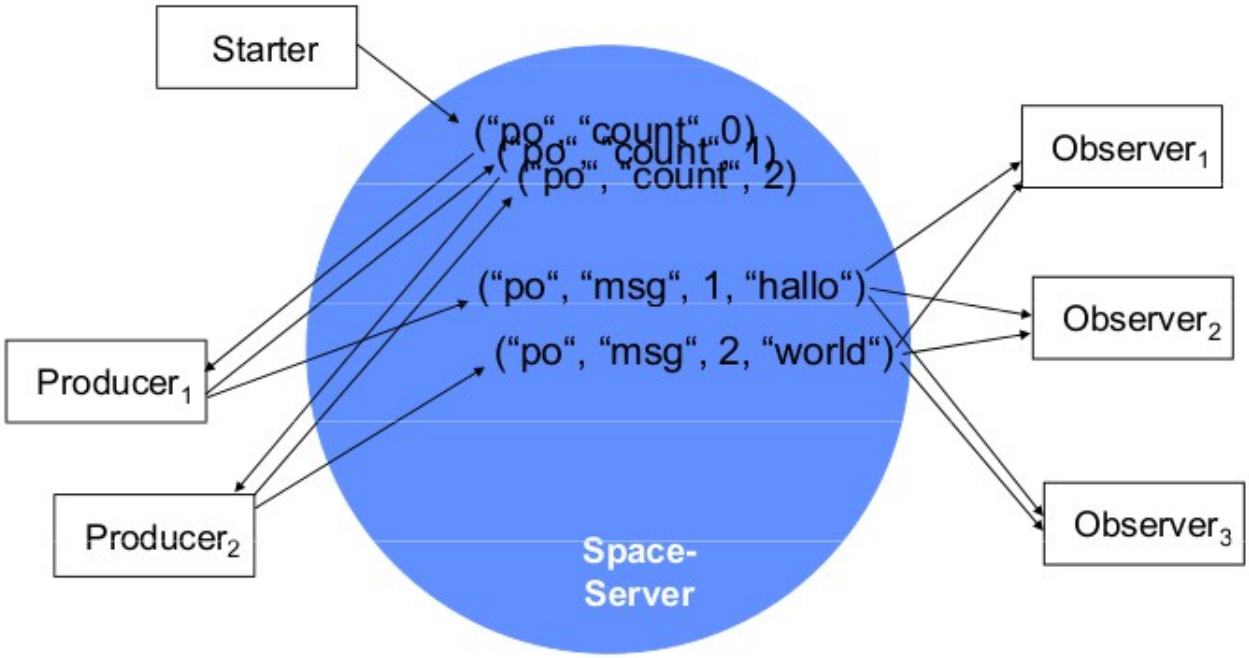
Welche vier Grundfunktionen spezifiziert der JavaSpaces Standard und was machen die Funktionen? (4)

- write(): Writes new objects into a space
- take(): Retrieves objects from a space
- read(): Makes a copy of objects in a space
- notify: Notifies a specified object when entries that match the given template are written into a space

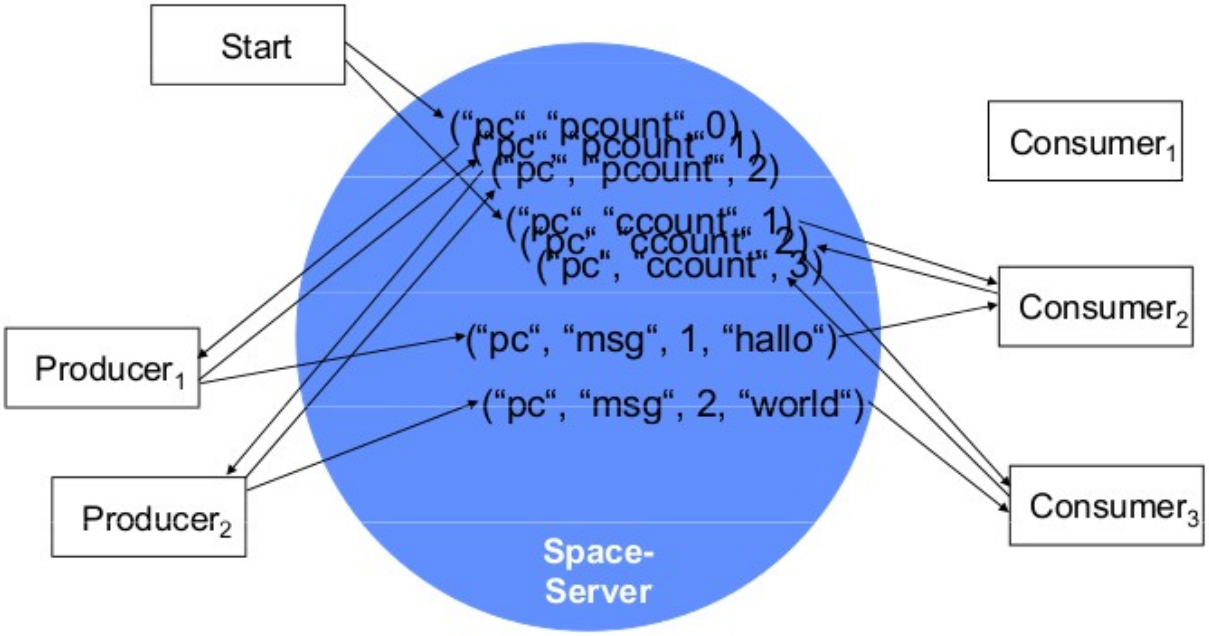
Koordinationsstruktur von Publish/Subscribe Pattern in JavaSpaces (Zeichnen/Erklären)



Skizzieren und beschreiben des Producer/Observer-Patterns mit JavaSpaces



Koordinationsstruktur von Producer und Cosumer bei Java Spaces



Aufzählen der 3 Hauptunterschiede von JavaSpaces zu XVSM (2)

JavaSpaces	**XVSM**
Verwaltet Entries (Tupels)	Mehere Koordinationsmodelle
	Aspekte
	Strukturierung des Koordinationsraumes
	Echte Peer to Peer Architektur
	Grundkonzept Container

Wie kann man die Bulk operationen write(List entries,...) und take(List templates,...) in JavaSpace05 Methoden implementieren

XVSM (eXtensible Virtual Shared Memory)

XVSM Konzepte:

- Sturkturierung des Koordinationsraumes
  - Shared Structured Containers
- Mehrere Koordinationsmodelle
  - Linda, FIFO, LIFO, KEY, LABEL
- Grundkonzept Container
  - bounden/unbounded
  - enthält Entries
- Core API (CAPIs) für den Zugriff
  - Container: create lookup, destroy, etc.
  - ContainerEntries: write, read, delete
- Flexible Plug-In Techniken (dynamische Erweiterbarkeit)
  - für alle CAPIs
  - auf Container oder Space Ebene

Nennen Sie 3 Coordination Types, die von XVSM unterstützt werden, und beschreiben Sie diese. (2)

- Query
- Linda
- FIFO
- LIFO
- Key
- Label
- Random/Any
- Vector

4 Methoden von Containern in XVSM nennen und beschreiben

Container Create

Container Lookup

Container Lock

Container Destroy

Was ist ein Container? Wie wird er erzeugt? Wofür ist hier die ContainerRef Klasse wichtig?

- Container: strukturierter Subspace, mit einer Vielzahl an Koordinationsmöglichkeiten, wie Linda, FIFO, LIFO, Label, Key etc.
  - bounded
  - named
  - enthält Entries
- über CAPI mit space Url und Koordinator: Capi.createContainer(Url, Koordinator)

Container-Reference:

- Mit Instanz von Capi und Methode mit Argument Container-Referenz erhält man Zugriff auf den Container – read, delete, take, write
- Repräsentiert und identifiziert einen Container
- Enthält auch die URL des Spaces in dem der Container ist

Unterschied zwischen write und shift, vor allem bei vollem Container?

Was ist der Unterschied zwischen read, take und delete in XVSM? (3)

read:

- The read method returns a number of Entries from the container, without deleting them.

take:

- This method is the same as first calling read and then delete: The read Entry is immediately deleted after reading in one atomic step. This functionality is also commonly referred to as “consuming read”. Also the take method could block and thus has a timeout parameter.

delete:

- Depending on the container’s Coordination type, an entry is deleted, but without returning the deleted entry. Note that also, when deleting entries, the method blocks until a suitable entry is found in the container.

Wie würde man einen Aspekt machen dass man auf einen Container Zugriff hat? (security, login)

vor container einen aspekt einfügen, der die security überwacht z.b. mit login authentifiziert.

Was sind Aspekte? Welche Typen gibt es? Auf welche Aktionen werden sie angehängt? (4)

flexible Plug-In Techniken für dynamische Erweiterbarkeit

- für alle CAPI-Operationen (create, destroy, lookup, lock, etc.), davor oder danach

– auf Container – oder Space-Ebene

z. B. Logging, Authentifizierung

„The idea behind Aspects is to have a hook before and after each space operation, for example a preWrite and a postWrite method. You implement these pre- and post-methods to perform some special action that is done automatically before or after you call one of the following methods of the Capi class. “

„For a new Aspect, you overwrite the pre- and post-
methods to add new functionality and then, you register the Aspect in the Capi. “

Unterschied Space(global)- und Container(lokal)-Aspekte:

Container-Aspekte betreffen einen bestimmten Container, Space Aspekte betreffen die gesamte Runtime → Alle Container bzw. globale Methoden

„Container Aspects are aspects that are connected to actions on a specific container, while Space Aspects refer to the space itself or to all containers. “

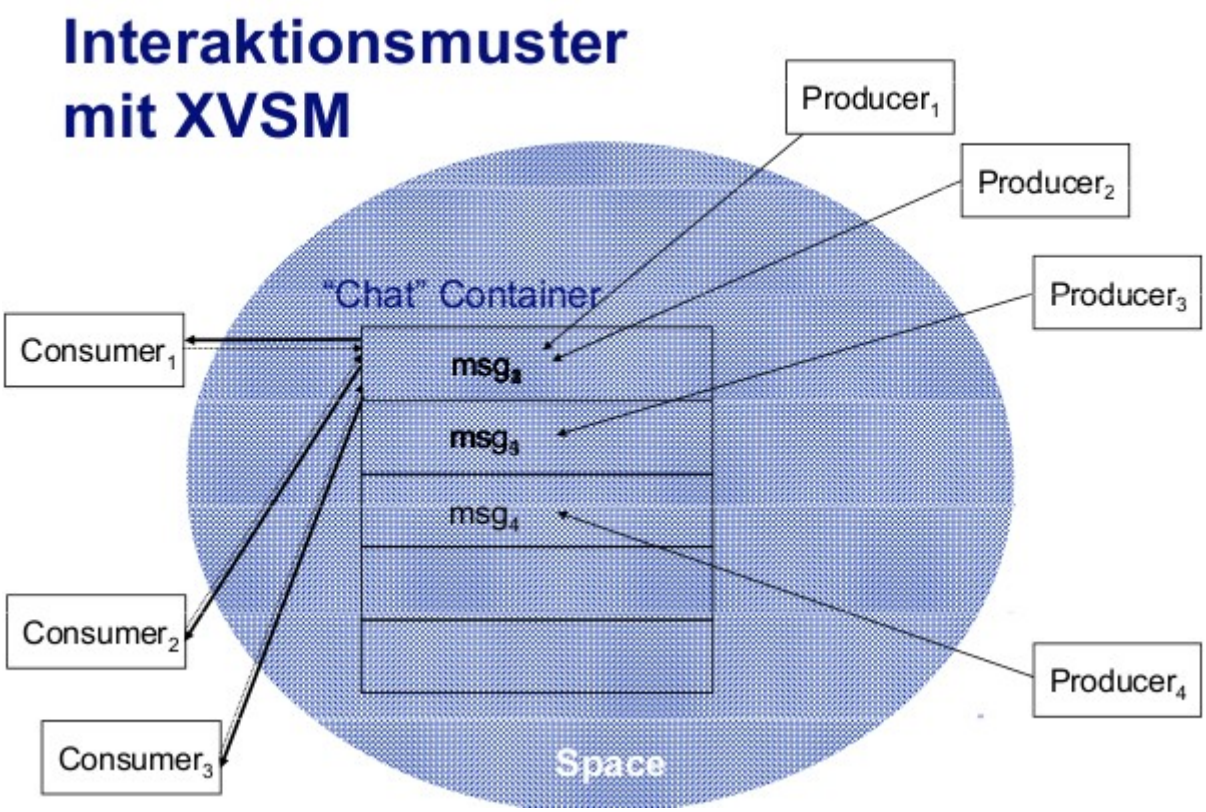
Unterschied sync Capi und async Capi:

Methoden wie bei synchroner Capi, aber:

- kein blockierendes Warten auf Ergebnis
- RequestFuture als Rückgabewert ODER
  - Methoden zum Pollen oder blockierenden Warten
  - Optional RequestCallback, Aufruf bei Ergebnis
- Answer-Container (Referenz als Argument)
- Ziel für Ergebnis
- Container im Space, lokal oder remote
- Optional: String als Key für Key-Coordinator

Unterschiede zwischen Atomic Entry und XVSM tupll (tupelspace??)

Producer/Consumer in XVSM beschreiben mit Skizze und Koordinationsstruktur



Erkläre Timeout bei XVSM read; Verhalten bei 0,inf. timeout bei read wenn

a) Eintrag vorhanden
b) Kein Eintrag vorhanden

„So giving a timeout of you have a non-blocking read method call – it either reads an entry or if there is none, it immediately returns an Exception. TRY_ONCE waits for locked Entries. If you don't want this behaviour use MzsConstants.RequestTimeout.ZERO. The MzsConstants.RequestTimeout.INFINITE, opposite would be to use the which waits forever, if no one ever writes a suitable entry to the container. "

Was sind Notifications? Für was werden sie definiert? (2)

„With notifications, you can listen on various actions and be called back if such an action is performed. For example: You have a listening class that would like to be notified if a write is called (and thus, a new entry is added) "

„A Notification is used if your class is interested in a special event in the Container, for example if an Entry was written or read. "

Beschreibe XVSM Logging für alle Operationen

Mit Aspekten vor und nach jedem Methodenaufruf eine Ausgabe machen?!

CORSO

Nennen Sie die Grundkonzepte von CORSO (2)

- Shared Objects mit einer netzwerkweit eindeutigen Identifikation (OID)
- Vollständige und symmetrische P2P Replikation zwischen identischen Peer-Knoten
- Verteiltes Flex-Transaktionsmanagement
- Starten und Überwachen von Prozessen
- Eingebaute Persistenz

Was ist in CORSO eine OID? Erklären sie den Unterschied. (2)

OID ist netzwerkweit eindeutige Identifikation

Welche 2 Typen von Kommunikations Objekten werden in Corso unterschieden?

- CONST: kann nur einen Wert zugewiesen bekommen
- VAR: Wert kann beliebig oft zugewiesen werden

Was ist eine Timestamp bei CORSO und wofür werden sie benutzt? (2)

- logische Zeitmarke

- unterschiedliche Art der Verwendung wird gesteuert:
  - CONST: kann nur einen Wert zugewiesen bekommen
  - VAR: Wert kann beliebig oft zugewiesen werden

Verteilungsfunktionen bei CORSO (2)

- EAGER : Jeder CoKe der eine Kopie des Objects besitzt, muss dessen Inhalt sofort aktualisieren.
- LAZY: Erst wenn ein CoKe von einem Prozess eine Anfrage zu diesem Object erhält, findet die Aktualisierung statt. Dies ist insbesondere bei sehr großen Daten von Vorteil, wenn sich deren Inhalt selten andert oder aber die Aktualität keine besondere Rolle spielt.
- RELIABILITY Klasse: 0, 1 oder 2

Nennen Sie 3 Koordinationstypen von CORSO

• Zuverlässige Kommunikation durch gemeinsam verwendete Datenobjekte.
• Nebenläufigkeit, Parallelität und Verteilung durch CORSO Prozesse.
• Flexible Koordinationspatterns und Wiederehrherstellbarkeit durch das hoch entwickelte Transaktionsmodell.

Welche Transaktionsstrategie benützt CORSO?

Optimistische Concurrency Control:

- keine Locks

- Hard & Soft Commit
- Request Cancellation

ACID

- Isloation Level:
  - Serializable: Die Wirkung parallel ablaufender Transaktionen ist exakt dieselbe, als ob die Transaktionen nacheinander in Folge ablaufen würden

ACD:

- Early Commit von Sub-Transaktionen (nested transactions)
- Semantic Compensation

Request/Answer in CORSO skizzieren und erklären.

Erkläre Named Objects in CORSO; Wo kann man sie sinnvoll einsetzen?

Um die Programmierung von verteilten Applikationen zu erleichtern, bietet Corso auch eine Art Name-Service an, womit einem bestimmten Communication Object innerhalb eines CoKe ein eindeutiger Name zugeordnet werden kann (Named Object).
Ein Prozess besitzt eine Referenz auf ein Objekt, wenn er entweder das Objekt kreiert hat, die Objektreferenz beim Prozessstart übergeben wurde oder das Objekt ein Subobjekt eines bereits bekannten Objektes ist. Das Konzept der „named objects“ erlaubt es Objekte innerhalb einer Domäne mit einem eindeutigen Namen zu identifizieren und zu referenzieren.

Was ist eine geeignete Koordinationsstruktur für das Producer/Observer-Pattern in CORSO (Zeichnung und Erklärung)

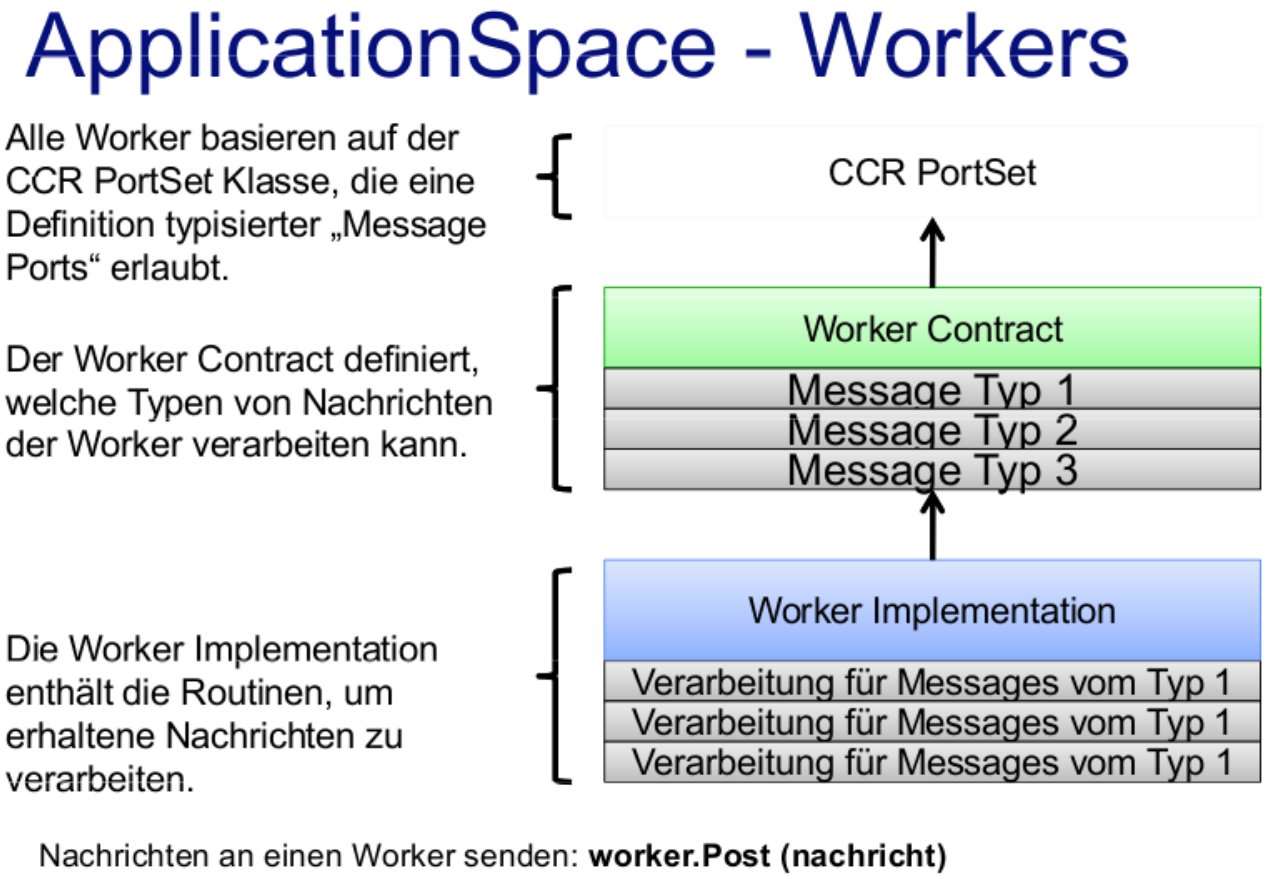
Skizzieren und beschreiben des Publish/Subscribe-Pattern mit CORSO(andere Gruppe mit JavaSpaces) (2)

Unterschiede von CORSO zu XVSM (die andere Gruppe hatte die Unterschiede von JavaSpaces zu XVSM) 2008

3 Konzepte von Application Spaces

Application Spaces: Hosting von Komponenten, sogenannten Workern

- lokale und verteilte Worker
- synchrone und asynchrone Worker



3 Konzepte von SOA (Service Oriented Architecture)

- Skalierbarkeit: Ob & wann, dynamisches Kommen und Gehen, sich selber koordinieren/orientieren
- Komplekitätsreduktion: kein 'allwissender' Koordinator/Vordenker, Zerlegung in kleinere autonome, überschaubare Einheiten
- Nicht planbare Situation: z.B. Wasserrohrbruch
- Sicherheit: ich bestimme ob und was ich tue
- Beliebige Kooperationsformen: nicht nur FIFO

P2P

Was versteht man unter P2P? Welche Kriterien soll es erfüllen? (2)

- Rechner im Internet bieten freistehende Ressourcen:
  - Inhalt (Files), CPU, Speicherplatz, Bandbreite
- Das P2P Paradigma ist ein natürliches, naheliegendes Konzept zur Organisation dieser Ressourcen, bei dem gleichberechtigte Rechner (Peers) im Internet
- miteinander kommunizieren
- Ausnutzen von Speicherkapazitäten auf Client Rechnern
  - wesentlich billigere Lösungen
- P2P ist die natürliche Ergänzung des zentralen Servers

Kriterien:

- Skalierbarkeit
- Performance
- Resistenz gegenüber Zensur
- Keine zentrale Kontrolle
- Verbesserter Zugriff auf verteilte Ressourcen
- Dynamische Anpassung an/Reaktion auf Fehler
- Dynamisches Kommen und Gehen von Peers

Unterschiede zwischen P2P und Client Server Modell

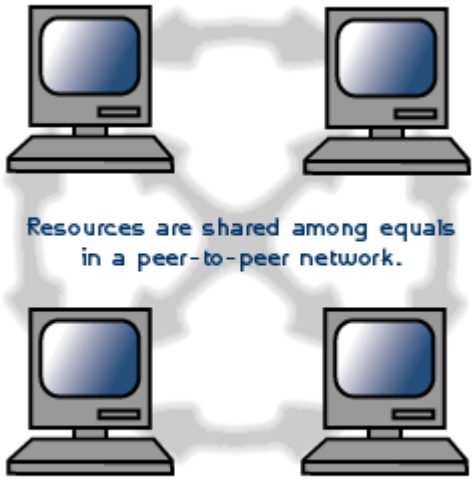


Fig. 1. Peer-to-peer network

Peer2Peer
Peer-to-peer network operating systems allow users to share resources and files located on their computers and to access shared resources found on other computers. However, they do not have a file server or a centralized management source (See fig. 1). In a peer-to-peer network, all computers are considered equal; they all have the same abilities to use the resources available on the network. Peer-to-peer networks are designed primarily for small to medium local area networks.

Client-Server Modell
Client/server network operating systems allow the network to centralize functions and applications in one or more dedicated file servers (See fig. 2). The file servers become the heart of the system, providing access to resources and providing security. Individual workstations (clients) have access to the resources available on the file servers. The network operating system provides the mechanism to integrate all the components of the

network and allow multiple users to simultaneously share the same resources irrespective of physical location.

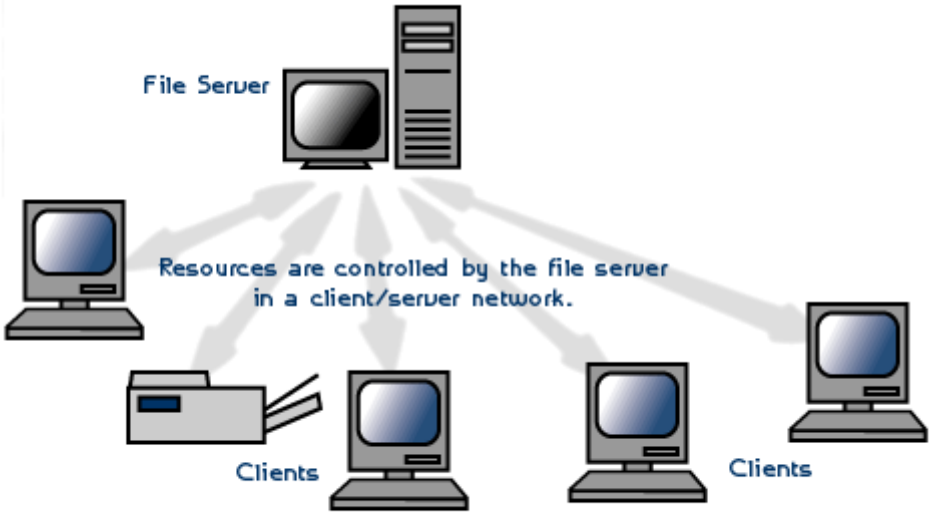


Fig. 2. Client/server network

3 Einsatzgebiete von P2Paufzählen und deren Vorteile in dem jeweiligen Gebiet erklären.

1. Universelles effizientes File Sharing
   - Datenhaltung vor Ort in Virtual Private Networks
   - Automatische Benachrichtigung bei Änderungen
2. (Mobile) E-Collaboration
   - Verteiltes, kooperatives Arbeiten (Teleworking, eCollaboration, kollaboratives Design, gemeinsam Dokumente bearbeiten etc.)
   - Ad hoc Arbeitsgruppen bilden
   - Multi user Spiele über das Internet
   - Web 2.0
3. Dynamische autonome Geschäftsprozesse
   - Ad hoc Workflows
   - Autonome Agenten

Vorteile von P2P

- Hohe Performance k ann durch Hinzufügen von billiger Hardware und Software erzielt werden (keine teure Server Hardware notwendig)
- Entlastung des Servers – wichtig für mobile electronic commerce (mobile smapr devices, die über Bluetooth und UMTS kommunizieren
- Privatsphäre der Daten kann gewahrt bleiben

Beschreiben Sie mögliche Probleme / Herausforderungen bei der Implementierung eines P2P-Systems. (3)

Technische Herausforderungen

- Sicherheit (Peers sind „untrusted“)
  - Speicherung, Routing, Access Control, Authentifizierung, Identitätsmanagement etc.
- Skalierbarkeit (bei Ansteigen der Peers Anfragen etc)
- Performance (Optimale Last- und Ressourcenverteilung)
- Fairness
- Ausfallsicherheit Verfügbarkeit, Fehlertoleranz durch Replikation (insb. in strukturierten P2P Systemen)
- Anonymität
- Verantwortung
- Belohnungssysteme („Incentive“)
- Komplexeres Daten-Management: Löschen, Update, Life Cycle Management / Expiration, Versionierung, komplexere Queries, (semantisches) Gruppieren von Daten

Wie funktioniert das P2P-Modell von Napster? Nennen Sie je einen Vor- und einen Nachteil. (2)

Strukturierter, hybrid zentralisierter P2P Dateiaustausch mit Serverunterstützung

Vorteile

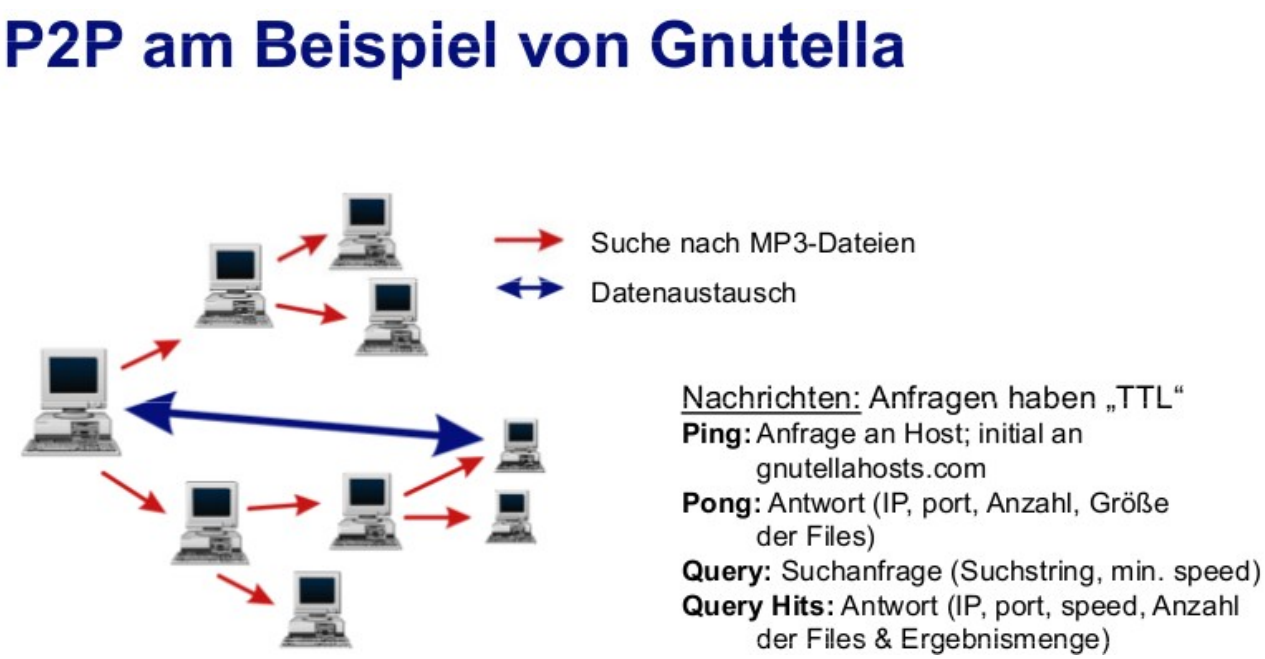
- Einfache Struktur daher einfache Eingriffsmöglichkeiten
- Ersparnis bei Bandbreite: kleine Datenmengen zw. Peer und Server, große Datenmengen zw. Peers

Nachteile

- Ein Server hat das Gesamtinhaltsverzeichnis: single point of failure
- Private Daten liegen auf externem Rechner
- Skaliert überhaupt nicht

Wie funktioniert das Gnutella-Netzwerk (Vor/Nachteile) (3-5)

Unstrukturierter, dezentraler P2P Dateiaustausch



Vorteile

- Kann sehr dynamisch sein
- Ad hoc Netzwerke sind möglich

Nachteile

- Der Hauptzweck, die Suche nach Dateien, führt zu hoher Netwerkbelastug durch (erwünschten) Schneeballeffekt
- Zerfall in Teilnetze, wenn ein Peer ausfällt
- Keine zentrale Kontrolle: niemand weiß, wer Teilnehmer ist
- Schlechte Skalierbarkeit (TTL → „virtueller Horizont“ pro Peer)

CAN beschreiben. Vor- und Nachteile. (2)

Strukturierte, dezentrale P2P Infrastrukturen

P2P am Beispiel von CAN

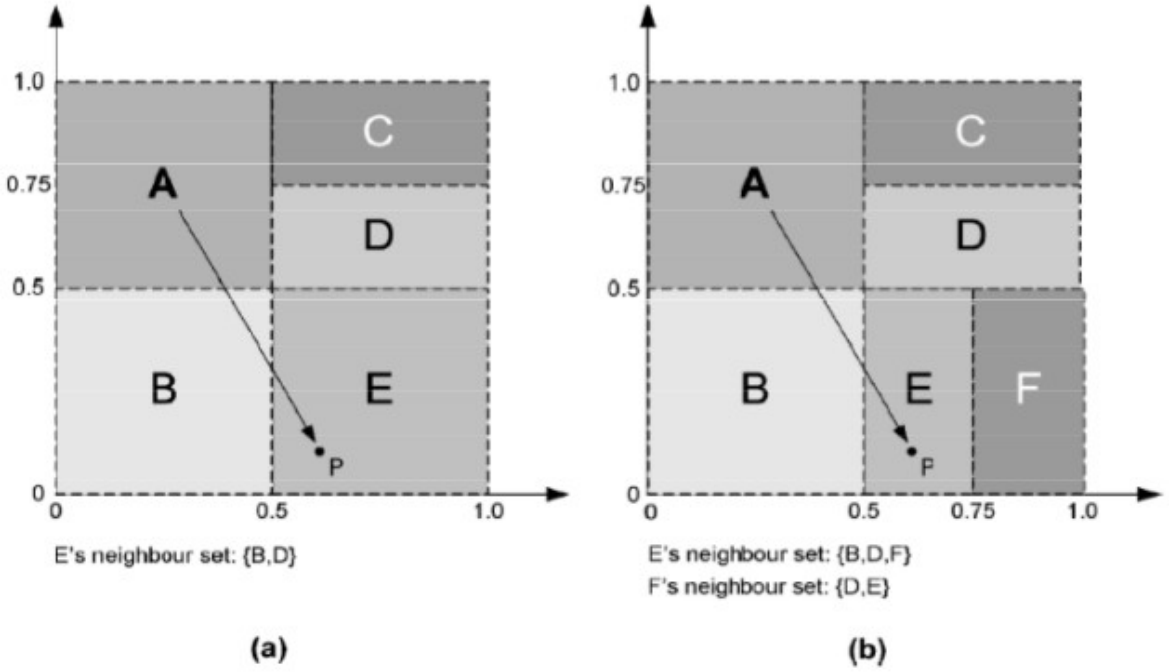


Fig. 6. CAN: (a) Example 2-d $[0,1] \times [0,1]$ coordinate space partitioned between 5 CAN nodes; (b) Example 2-d space after node F joins.

Vorteile

- Skalierbarkeit
- Einfache deterministische (Key Value) Abbildung in den Space
- Jeder Knoten muss nur seine Zone (Teil des Hash Table) und wenige Nachbarn verwalten („Routing Table“)

Nachteile

- Dynamisches Kommen & Gehen von Knoten: aufwändig
  - JOIN Zone aufteilen; Routing Table von neuem und Nachbarknoten updaten;
  - analog: kontrolliertes LEAVE: Handover an Nachbarn
- Verfügbarkeit, wenn viele Knoten glz. ausfallen
- Nur „exact match“ Lookup möglich

Erkläre Cycle Sharing SETI@Home. Wieso ist es kein echtes P2P? (2)

Unstrukturiert und völlig zentralisiert:
Cycle Sharing – Nutzung von Rechenleistung, technisch kein P2P
Kann mit Napster-Modell zu echtem P2P ausgebaut werden

Vorteile

- Gigantische Rechenleistung

Nachteile

- Beim Ausfall des Servers stehen alle Clients
- Work-Units müssen voneinander unabhängig sein
- Was passiert, wenn ein Client kein Ergebnis liefert? Wie lange soll man warten?
- Eignet sich nur für bestimmte Rechenvorgänge

Probleme Verteilter Applikationen - 3 Beispiele (2)

Erklärung von „Reliability“ und „Transactions“ (bzw. „Load Balancing“ und „Replication“

Location - Explizite Adressierung von Ressourcen via Namen
Migration - Ressourcenverteilung erfordert Namensänderung
Replication - Administration von verteilten Datenkopien
Access - Zugriff auf lokale / verteilte Ressourcen ist unterschiedlich
Transactions - Garantie von transaktionaler Sicherheit
Reliability - Fehlerbehandlung durch Funktionsreplikation
Representation - Unterschiede betreffend Sprache & Darstellung von Daten
Load Balancing - Automatische Behandlung unterschiedlicher Systemlasten
Recovery - Automatischer Wiederanstart von Kommunikationspartnern

2 Nachteile von Message Queues (2)

- in queues the data flow is directed between end points
- spaces represent a shared state space
  - coordination data structures offer arbitrary access
  - represent the entire state of a business process
- „read once“ versus „read many“

Nennen Sie 4 der 8 Trugschlüsse verteilter Kommunikation (3)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

Aufzählen der 3 Erwartungen für GRID (3)

- autonome Komponenten (Peers), die in der Lage sind
  - ihr Zusammenspiel zu organisieren und
  - sich selber zu heilen (self-* Eigenschaften)
- flexible Zuordnung (on-demand)
  - von verfügbaren (schlafenden) Ressourcen (CPU, Memory, Netzwerk)
  - entsprechend der verlangten Last
- Management of SLAs (Service Level Agreements)
  - Monitoring, Security, Auditing, Billing, etc.

Was ist ein Overlay Network und in welche Kategorien kann man es einteilen? Nennen Sie ein konkretes Beispiel fuer ein System oder eine Infrastruktur und geben Sie an, in welche Kategorie es fällt.

Foliensatz I Seite 17

Der Begriff Overlay-Netzwerk beschreibt ein Netzwerk, das auf ein bestehendes Netzwerk aufsetzt (das sogenannte Underlay). Hauptmerkmale eines Overlay-Netzwerks sind:

- (logisches) Netz oberhalb existierender Infrastruktur
- oftmals eigener Adressraum mit eigener Adressierung (unabhängig vom Underlay)
- ggf. Einsatz eigener Wegewahlverfahren

Das Overlay-Netzwerk wird daher zum Aufbau einer zusätzlichen Topologie (physikalisch, logisch, strukturell,...) genutzt. Mit der neu entstandenen Struktur lassen sich beispielsweise aktuelle Technologien wie IPv6 testen oder alternative Routen für die Daten anlegen. Peer-to-Peer-Netze, wie zum Beispiel Gnutella und Chord sind ebenfalls auf einer Overlay-Netzwerkstruktur aufgebaut, über die beispielsweise effizientes Suchen und eine gewisse Netzwerkkoordination erst möglich werden. Auch auf diese Overlays können weitere Overlays aufsetzen, z.B. für das Monitoring des P2P Overlays wird SkyEye genutzt, das einen eigenen Adressraum einführt und eigene Protokolle mitbringt, die die Funktionalität der P2P Overlays voraussetzen.

Das Internet war in seinen Anfängen ein Overlay-Netzwerk über dem bestehenden Telefonnetz.

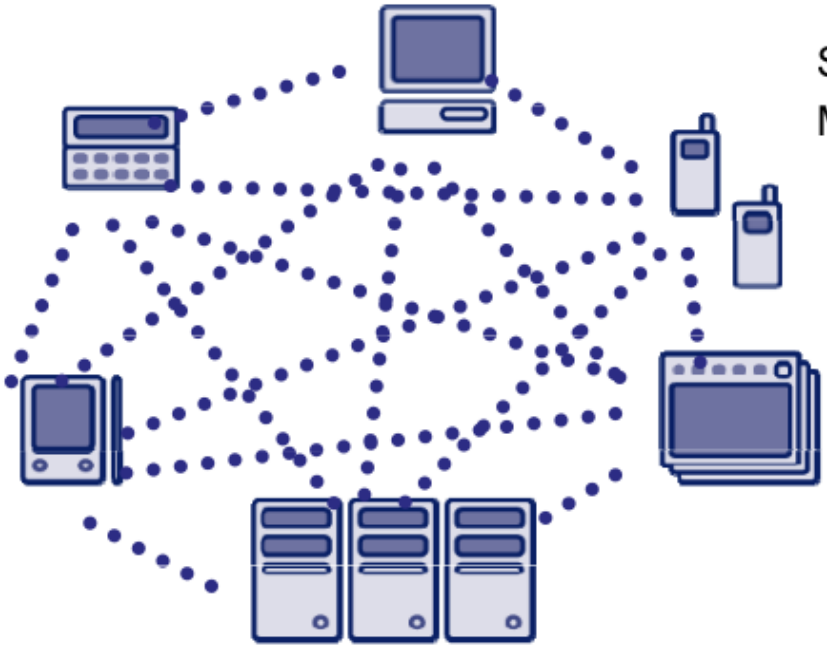
Nennen Sie 2 Vorteile von Space Based Computing gegenüber herkömmlichen Message Passing Architekturen. (3)

selektive Auswahl durch Koordinatoren
organisiert
topic t

Aufzählen der 3 Evolutionsstufen von Middleware + Beispiele der jeweiligen Vertreter (3)

Middleware: Evolutionsstufe 1

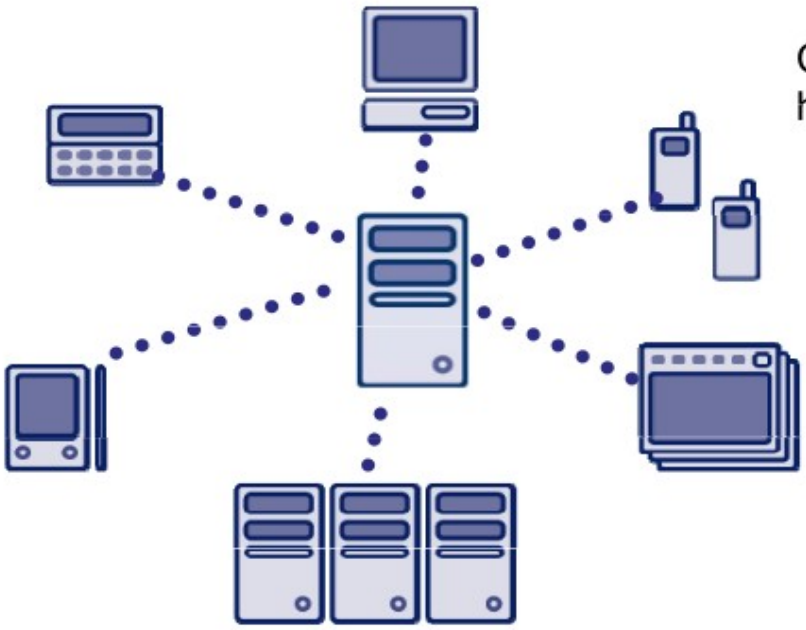
Direkte Kommunikation / Message Passing



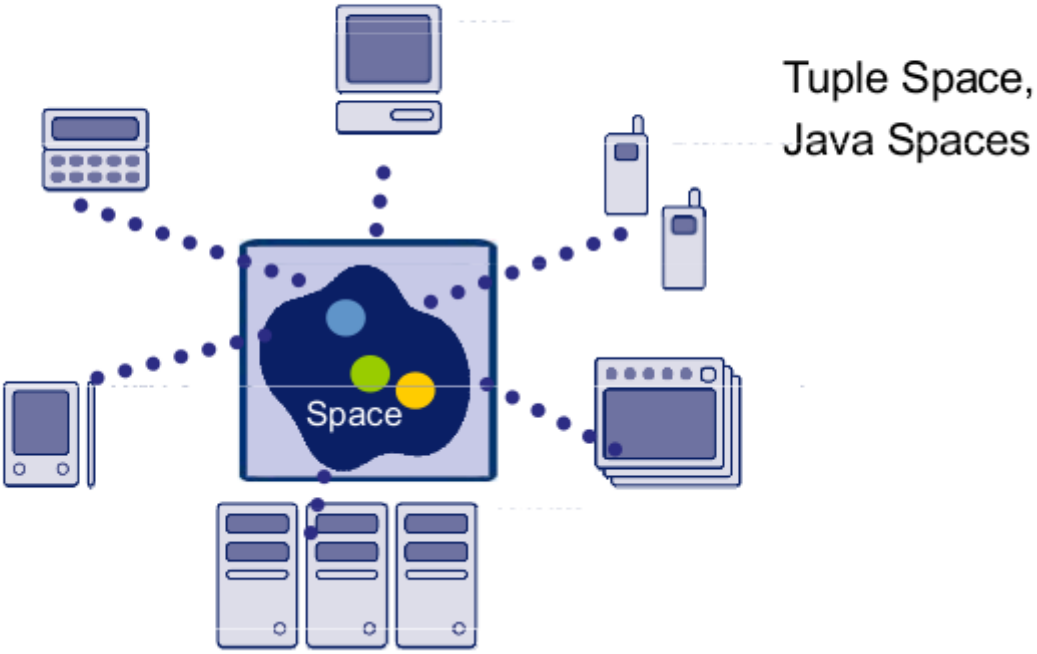
Sockets, RPC, MOM, RMI

Middleware: Evolutionsstufe 2

Zentraler Server / Message Passing



Client/Server, http

Middleware: Evolutionsstufe 2-3

Zentraler Server / Space Based Computing