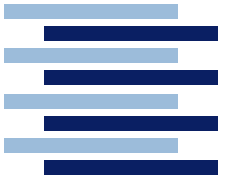
Netzwerk-transparente Persistenz

Lutz Behnke
09.06.05

Kai von Luck

Überblick

- Persistenz allgemein
- Zieldefinition
- State of the Art
  - Remote Filesystem
  - (verteilte) Datenbanken
  - ORB-Persistenz
  - Peer2Peer
- Vorschlag für Projekt
  - Motivation
  - Schnittstelle



Definition Persistenz

Persistenz ist die Eigenschaft eines Objektes durch die seine Existenz eine bestimmte Zeit überdauern kann (d.h. das Objekt existiert über die Existenz seines Erzeugers hinaus) und/oder (Namens-)Raum durchdringt (d.h. der Ort des des Objektes entfernt sich vom Namens- oder Adressraum in dem es erschaffen wurde).

(Grady Booch [1])

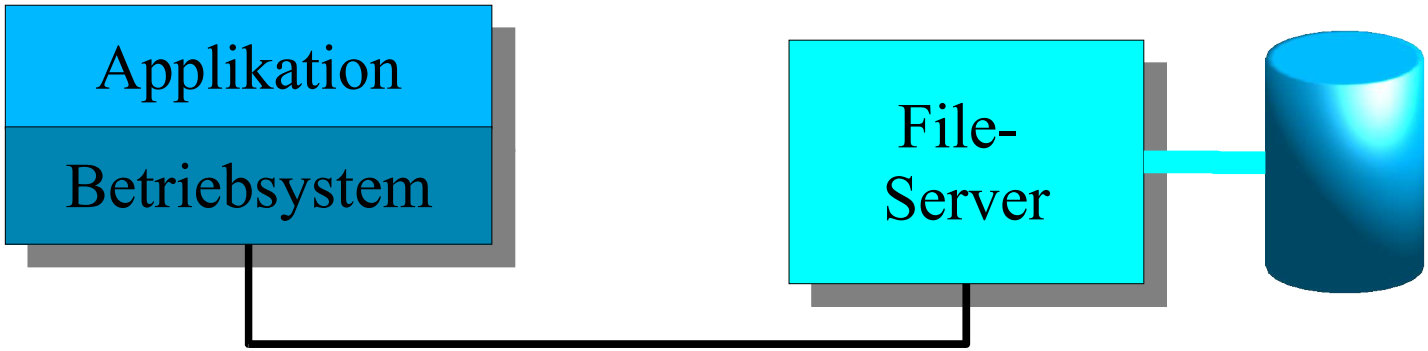
Netzwerk-Persistenz ist erweitert diese Eigenschaft um die Fähigkeit zum richtigen Zeitpunkt die richtigen Objekte an der richtigen physikalischen Position vorzuhalten.

(Lutz Behnke)

Zieldefinition

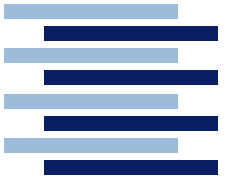
- Benutzer braucht Speicherort nicht kennen
- Dienst ist immer und überall verfügbar.
  - Nicht nur im eigenen Netz
  - Ad-Hoc: keine individuelle Konfiguration von Komponenten
  - Sicher (safe) und sicher (secure)
  - möglichst viel Arbeit soll durch System übernommen werden.
- Transparent für Entwickler von Anwendungen
  - egal ob auf Client oder Server
- hohe Skalierbarkeit: Betrieb wie ein Handy-Anbieter
  - >100K angeschlossene Computer (bei n:1 Computer:Benutzer)

Netzwerk Persistenz: Remote Filesystem (I)



- Server-zentriert
  - Server entscheidet über Berechtigungen und verwaltet Identitäten.
  - Jeder User muß dem Server bekannt sein. Bei mehreren Server muß es entweder einheitliche User-Kennungen geben oder der User muß sich viele Username/PW merken -> multiple Websites.
- Applikation muß Ablage-Strategie selbst implementieren.
- Daten liegen nur auf Server (Wenn man Caching ignoriert)
- Typische Vertreter: NFS, SMB

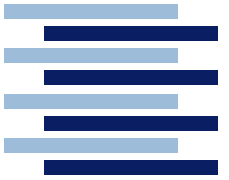
Objekt Operationen (I)



- Create: Anlegen von neuen Objekten
- Read: Lesen von Objekten
- Change: Ändern von Objekten
- Delete: Löschen von Objekten

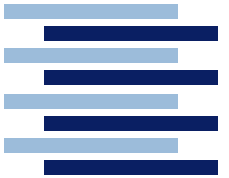
Zugriffs-Negotiation wird durch exklusive Locks geregelt. Keine Notwendigkeit für Transaktionen auf Dateisystem-Schnittstellen-Ebene, da hier ohnehin kein Wissen über die Strukturen der Objekte vorhanden ist.

Netzwerk Persistenz: Remote Filesystem(II)



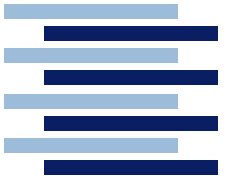
- Probleme
  - Immer Online
  - Keine Hilfe bei Objekt-Strukturen
  - Der Server ist Single-Point-Of-Failure
  - Skaliert nicht gut.
- Wege weg vom einzelnen Server:
  - Storage Area Network (SAN): gfs
    - Trennung Speicherung und Betriebsystem
  - Server-Cluster: lustre, googleFS
    - Trennung Daten- und Meta-Daten
  - Offline: Coda
  - Andrew Filesystem (AFS)

Remote Filesystem: Coda



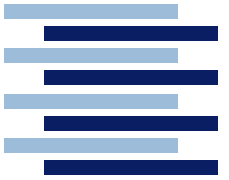
- Clients können On- oder Offline arbeiten
- Daten werden zwischen Servern aufgeteilt und gecached
- Nachteile
  - Um Änderungen in Dateien zusammenzuführen sind Format-spezifische Plug-Ins notwendig.

Objekt Operationen (II)



- Create: Anlegen von neuen Objekten
- Read: Lesen von Objekten
- Change: Ändern von Objekten
- Delete: Löschen von Objekten
  - Online wurde noch geändert während Offline gelöscht wurde. Was hat priorität?
- Merge
  - Änderungen an Objekt A und A' in A'' zusammenführen.

Remote Filsystems: Trennung von Daten und Metadaten

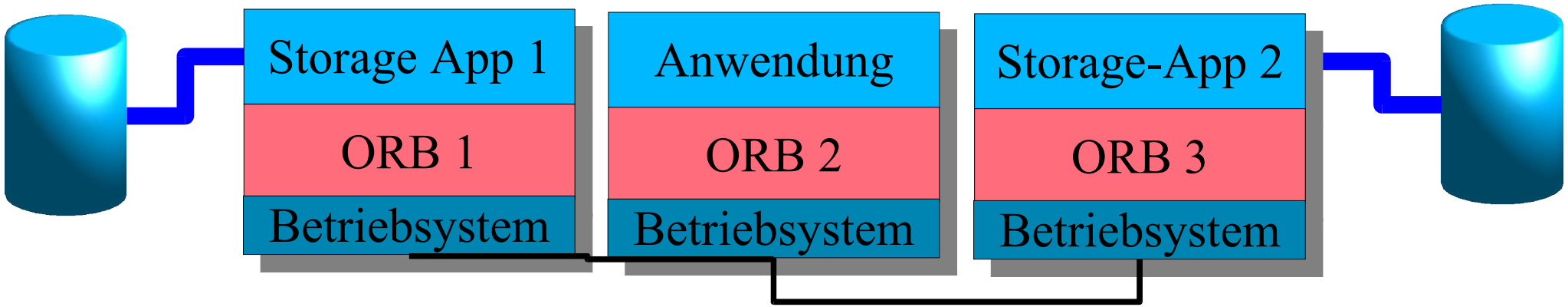


- Mehrere Server um Last zu verteilen
- (sehr) viele Datenserver
- ein/wenige Metadatenserver
- Beispiele
  - lustre
    - Mehr Intelligenz in die Festplatten, aber keine Dateisysteme
  - GoogleFS
    - Beispiel: 2 Master, 227 Chunkserver, 155 TB Daten
    - Voll-replizierte Master um PoF zu eliminieren

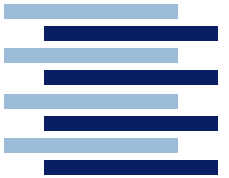
Network Persistence: Database

- Server übernimmt Ablage Strategie
- Server enthält Daten-Schema
- Applikation bereitet Objekte zur Ablage vor.
- Verteilte Datenbank trennen virtuellen Server von physikalischer Maschine.

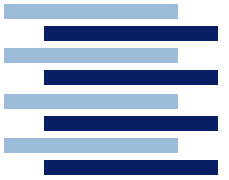
Formen der Netzwerk-Persistenz: Objekte über RMI/ORB



- Persistenz in Java
  - über Standard Java Persistenz Mechanismen (z.B. Hibernate, etc.) aber die Methoden werden per RMI aufgerufen.
  - Sprachspezifisch.
- Persistenz in CORBA-ORB
  - Dienst der Persistenz leistet ***kann*** entfernt sein. Die Anwendung weiß es aber nicht
  - Dienstleister können per abstraktem Namen gefunden werden.

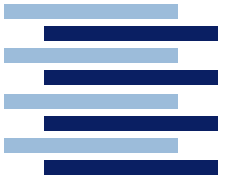


- Objekte werden Persistence State Definition Language (PSDL) definiert
  - Superset von IDL
- Probleme
  - Kein allgemein gültiger Weg einen Dienst zu identifizieren. (OMG versucht alle Tauben zu treffen)
  - Anwendung muß Abhängigkeiten zwischen komplexen Objekten implementieren.



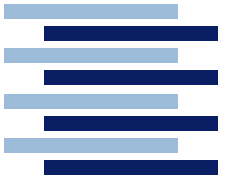
- Keine Feste Bindung Server<->Objekt mehr.
- keine hierarchische Organisation
- Teilnehmer (Nodes) können sich zu jedem Zeitpunkt in das System integrieren oder es verlassen.
- distributed object search
  - Objekte werden durch Suchen gefunden, nicht durch hierarchische Ablage
- 'read-only' persistence system
- Vorteile:
  - keine Absprache zwischen Knoten
- Nachteile:
  - keine Garantie das Objekte im System erhalten bleiben.

Proxy Remote Operationen



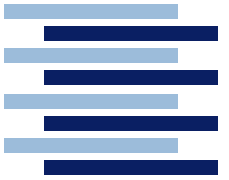
- Kopieren von Host B nach Host C ohne über Host A zu gehen
- Datei-Manipulation ohne Datei zu laden.
- Transparente Redundanz

Suchen mit erweiterten Metadaten



- Modell von GMail 
- Objekte nicht in komplexen Hierarchien ablegen
  - Hierarchien sind im allgemeinen Benutzer-zentrisch
- Suchen auf Metadaten
- Typische Metadaten
  - Generisch
    - Erstellungsdatum
    - Name (können mehrere sein)
  - Objekt-Spezifisch
    - Bildgröße
    - Durchschnittliches Rating meiner Freunde (z.B. Musikstücke)

Anforderungen



- Authorisation
  - Non-Repudiation
  - Vertraulichkeit
  - Unveränderlichkeit (Außer durch Berechtigte Personen)
- Change-Management
  - Variation des Split-Brain Problems
  - Transaktionen
- Verantwortung für Objekt Speicherung
  - Replikation
- ???

Angebot: Ich kümmere mich um den Server für die Ablage

- Heute:
  - Angabe des FTP/Datenbank Servers der die Daten entgegen nimmt.
- Morgen:
  - Software ermittelt wo Datenobjekte hin müssen und sorgt dafür.
  - Das „System“ liefert Datenobjekte auf Zuruf wieder zurück
    - Vorbild: DNS: wenig Zentrale Anlaufstellen, verteilte Datenbasis, (fast) immer verfügbar

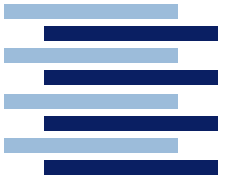
Umdrehen des Spießes: Service Oriented Architecture (SOA)

- Service Discovery Protocol um Server im Netz zu finden
  - Nicht vom Benutzer angegeben
- „Irgendwer wird mein Objekt schon annehmen“
- Benötigt natürlich (nahezu) perfektes Sicherheitsmodell
  - PDA erzwingt sparsamen Umgang mit asymetrischer Krypto.

Fehlerzustände

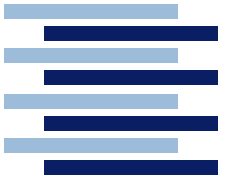
- Unvollständige Übertragung
  - Auf direkten Netzverbindungen hilft TCP
  - Schwierig bei Store-and-Forward Verbindungen
- Objekt kann nicht wiedergefunden werden
  - Verantwortung für Speicherung muss geklärt sein.
- Objekte von unterschiedlichen Subjekten geändert.
  - Wie ist ein Offline-Betrieb möglich?

Organisatorische Probleme



- Die eigentliche Schwierigkeit von Grids
  - verschiedene Organisation
  - verschiedene Personen
  - Abbildung von Vertrauen

Transparente Network Persistence (IDL)



- Abstrakte Schnittstelle für Objekt-Handling

```
interface Persistence {
        struct genericObject {
                long owner;
                long time;
                octet[]      data;
                };

    typedef octet[160] OpaqueKey;

        exception InvalidKeyException{};

        OpaqueKey registerObj(in genericObject obj);

        void retrieveObj(in OpaqueKey key, genericObject obj)
                raises InvalidKeyException;
};

interface AbstractGenerator {
        void retrieveAbstract(in OpaqueKey key)
                raises      InvalidKeyException,
```

Bibliographie

[1] Grady Booch: Objektorientierte Analyse und Design mit Praktischen Anwendungsbeispielen, 1995, Addison-Wesley, Bonn et.al.