The OceanStore Project

An Architecture for Global-Scale Persistent Storage

Johann Latocha

Universität Freiburg - Institut für Informatik

1. März 2007

Einleitung

Dateisysteme

1. Lokale Dateisysteme
   - Bsp. FAT32, NTFS, ext2/3
2. Verteilte Dateisysteme
   - Remote Dateisysteme
   - Cluster Dateisysteme
   - **Globale Dateisysteme**

Einleitung

Dateisysteme

1. Lokale Dateisysteme
   - Bsp. FAT32, NTFS, ext2/3
2. Verteilte Dateisysteme
   - Remote Dateisysteme
   - Cluster Dateisysteme
   - **Globale Dateisysteme**

Gegenstand

Was ist OceanStore?

Grundidee:

,,Serves in the world from an “ocean” of data.
This data quickly “flows” to where it is needed.“

Globales Dateisystem:

- University of California, Berkeley
- http://oceanstore.cs.berkeley.edu

Gegenstand

Was ist OceanStore?

Grundidee:

> ,,Serves in the world from an “ocean” of data.
> This data quickly “flows” to where it is needed.“

Globales Dateisystem:

- University of California, Berkeley
- http://oceanstore.cs.berkeley.edu

Vorteile

1. Hohe Speicherkapazität
2. Höhere Verfügbarkeit
3. Effiziente und dauerhafte Archivierung
4. Zugriff von jeder Stelle des Netzwerks
5. Betrieb durch verschiedene Organisationen

Vorteile

1. Hohe Speicherkapazität
2. Höhere Verfügbarkeit
3. Effiziente und dauerhafte Archivierung
4. Zugriff von jeder Stelle des Netzwerks
5. Betrieb durch verschiedene Organisationen

Vorteile

1. Hohe Speicherkapazität
2. Höhere Verfügbarkeit
3. Effiziente und dauerhafte Archivierung
4. Zugriff von jeder Stelle des Netzwerks
5. Betrieb durch verschiedene Organisationen

Vorteile

1. Hohe Speicherkapazität
2. Höhere Verfügbarkeit
3. Effiziente und dauerhafte Archivierung
4. Zugriff von jeder Stelle des Netzwerks
5. Betrieb durch verschiedene Organisationen

Vorteile

1. Hohe Speicherkapazität
2. Höhere Verfügbarkeit
3. Effiziente und dauerhafte Archivierung
4. Zugriff von jeder Stelle des Netzwerks
5. Betrieb durch verschiedene Organisationen

Einsatzgebiete

Größenordnung (Schätzung)

- 10^{10} Benutzer
- 10000 Dateien pro Benutzer

Anwendung

- Digitale Bibliotheken
- Groupware-Dienste

Einsatzgebiete

Größenordnung (Schätzung)

- 10^{10} Benutzer
- 10000 Dateien pro Benutzer

Anwendung

- Digitale Bibliotheken
- Groupware-Dienste

Probleme

1. Ausfälle
2. Datendiebstahl
3. Effizienz

Kernkonzept

Zwei zentrale Aspekte

1. Sicherheit
2. Verfügbarkeit

Kernkonzept

Sicherheit

Szenario:

- Das Internet wird als Medium benutzt
- Einsatz in unsicheren Umgebungen

Lösung:

- Verwendung von Kryptographie
- Gebrauch von „floating replicas“
- Deep Archival Storage

Kernkonzept

Sicherheit

Szenario:

- Das Internet wird als Medium benutzt
- Einsatz in unsicheren Umgebungen

Lösung:

- Verwendung von Kryptographie
- Gebrauch von „floating replicas“
- Deep Archival Storage

Kernkonzept

Verfügbarkeit

Daten und Speicherort trennen:

- ,,nomadic data“

Effizienz

- ,,promiscuous caching“
- ,,introspective monitoring“

Kernkonzept

Verfügbarkeit

Daten und Speicherort trennen:

- ,,nomadic data“

Effizienz

- ,,promiscuous caching“
- ,,introspective monitoring“

Datenobjekt (1)

- Änderungen möglich
- Unterstützung von Zugriffskontrollen
- Versionskontrolle
- Verschlüsselung

Besteht aus:

1. Inhalt
2. GUID (Hashwert)
   - Dateiname
   - Öffentlicher Schlüssel

Datenobjekt (1)

- Änderungen möglich
- Unterstützung von Zugriffskontrollen
- Versionskontrolle
- Verschlüsselung

Besteht aus:

1. Inhalt
2. GUID (Hashwert)
   - Dateiname
   - Öffentlicher Schlüssel

Datenobjekt (2)

Objekte werden als Blockfragmente gespeichert:

- Organisiert durch B-Bäume
- Logarithmische Zugriffszeit
- Wurzelblock mit Metainformationen

Datenobjekt (3)

Versehen mit Markierungen:

1. Aktiv
   - Schreibzugriff
   - Am Stück vorhanden
   - **Im inneren Ring**
2. Archiv
   - Nur Lesezugriff
   - Stark fragmentiert
   - **Im äußeren Ring**

Datenobjekt (3)

Versehen mit Markierungen:

1. Aktiv
   - Schreibzugriff
   - Am Stück vorhanden
   - **Im inneren Ring**
2. Archiv
   - Nur Lesezugriff
   - Stark fragmentiert
   - **Im äußeren Ring**

Aktives Datenobjekt

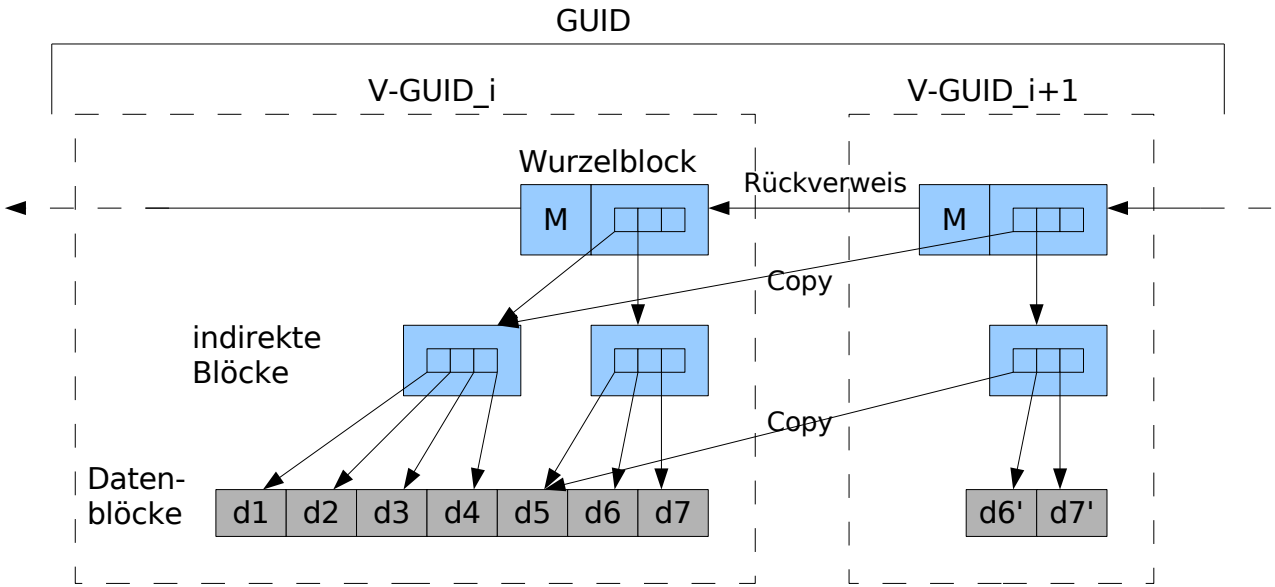

Abbildung: Aktives Datenobjekt [1]

Zugriffskontrolle

Durch PublicKey-Verfahren realisiert:

1. Lesezugriff
   - Muss Schlüssel besitzen
2. Schreibzugriff
   - Nur auf aktuelle Version
   - Signaturen der Autoren

Zugriffskontrolle

Durch PublicKey-Verfahren realisiert:

1. Lesezugriff
   - Muss Schlüssel besitzen
2. Schreibzugriff
   - Nur auf aktuelle Version
   - Signaturen der Autoren

Merkmale

1. Authentifikation des Clients
2. Mögliche Operationen auf Blöcke:
   - Löschen
   - Ersetzen
   - Anhängen
3. Alle Operationen auf verschlüsselten Blöcken
4. **Byzantinisches Vereinbarungsprotokoll**

Merkmale

1. Authentifikation des Clients
2. Mögliche Operationen auf Blöcke:
   - Löschen
   - Ersetzen
   - Anhängen
3. Alle Operationen auf verschlüsselten Blöcken
4. Byzantinisches Vereinbarungsprotokoll

Merkmale

1. Authentifikation des Clients
2. Mögliche Operationen auf Blöcke:
   - Löschen
   - Ersetzen
   - Anhängen
3. Alle Operationen auf verschlüsselten Blöcken
4. Byzantinisches Vereinbarungsprotokoll

Merkmale

1. Authentifikation des Clients
2. Mögliche Operationen auf Blöcke:
   - Löschen
   - Ersetzen
   - Anhängen
3. Alle Operationen auf verschlüsselten Blöcken
4. **Byzantinisches Vereinbarungsprotokoll**

Ablauf

1. Update nur auf neuen Block
2. Neue GUID (neue Version) wird erzeugt
3. Block und GUID werden an den inneren Ring übergeben
4. Alter Block: vom inneren Ring an äußeren Ring

Ablauf

1. Update nur auf neuen Block
2. Neue GUID (neue Version) wird erzeugt
3. Block und GUID werden an den inneren Ring übergeben
4. Alter Block: vom inneren Ring an äußeren Ring

Ablauf

1. Update nur auf neuen Block
2. Neue GUID (neue Version) wird erzeugt
3. Block und GUID werden an den inneren Ring übergeben
4. Alter Block: vom inneren Ring an äußeren Ring

Ablauf

1. Update nur auf neuen Block
2. Neue GUID (neue Version) wird erzeugt
3. Block und GUID werden an den inneren Ring übergeben
4. Alter Block: vom inneren Ring an äußeren Ring

Byzantinisches Vereinbarungsprotokoll

Mehr als zwei Drittel der Server müssen vertrauenswürdig sein, um einen Einfluss der ,,Lügner“ auszuschließen.

- Beispiel (n Server):
  Für $n = 3m + 1$ müssen mehr als m Server ,,böse“ sein.

- Nachteil:
  Sehr kommunikationsintensiv
  (Bei n Rechnern 2^2 Nachrichten notwendig)

Byzantinisches Vereinbarungsprotokoll

Mehr als zwei Drittel der Server müssen vertrauenswürdig sein, um einen Einfluss der ,,Lügner“ auszuschließen.

- Beispiel (n Server):
  Für $n = 3m + 1$ müssen mehr als m Server ,,böse“ sein.

- Nachteil:
  Sehr kommunikationsintensiv
  (Bei n Rechnern 2^2 Nachrichten notwendig)

Deep Archival Storage

Realisiert durch Sonderkodierung:

- Verwendung von **„Erasure Codes“**
  - Gleicher Speicherverbrauch, höhere Fehlertoleranz
    $m :=$ Anzahl Fragmente eines Blocks
    $n :=$ neue Anzahl Fragmente $(n > m)$
    $r = \frac{m}{n}$ Kodierungsrate
    Speicherbedarf um Faktor $\frac{1}{r}$ erhöht
  - Nachteil: Sehr rechenaufwändig

Eigenschaften

1. Deterministisches Auffinden
2. Lokalität des Routings
3. Minimalität und Lastenunterteilung
4. Dynamische Mitgliedschaft

Eigenschaften

1. Deterministisches Auffinden
2. Lokalität des Routings
3. Minimalität und Lastenunterteilung
4. Dynamische Mitgliedschaft

Eigenschaften

1. Deterministisches Auffinden
2. Lokalität des Routings
3. Minimalität und Lastenunterteilung
4. Dynamische Mitgliedschaft

Eigenschaften

1. Deterministisches Auffinden
2. Lokalität des Routings
3. Minimalität und Lastenunterteilung
4. Dynamische Mitgliedschaft

Lokalisationsmechanismen

1. Stochastischer Ansatz
   - Schnell
   - Lokal
   - Wird zuerst versucht
2. Deterministischer Ansatz
   - Strukturiertes Vorgehen
   - Global
   - Nach Scheitern von stochastischem Ansatz

Lokalisationsmechanismen

1. Stochastischer Ansatz
   - Schnell
   - Lokal
   - Wird zuerst versucht
2. Deterministischer Ansatz
   - Strukturiertes Vorgehen
   - Global
   - Nach Scheitern von stochastischem Ansatz

Plaxton Schema

Die Grundlage

1. Knoten bekommt **node-ID** fester Länge (zB. B4F8)
2. Knoten enthält einen Wurzelblock
3. Größtmögliche ID-Übereinstimmung entscheidet
4. **Neighbor-Map** in Level unterteilt
5. Für statische Netze

Plaxton Schema

Die Grundlage

1. Knoten bekommt **node-ID** fester Länge (zB. B4F8)
2. Knoten enthält einen Wurzelblock
3. Größtmögliche ID-Übereinstimmung entscheidet
4. **Neighbor-Map** in Level unterteilt
5. Für statische Netze

Plaxton Schema

Die Grundlage

1. Knoten bekommt **node-ID** fester Länge (zB. B4F8)
2. Knoten enthält einen Wurzelblock
3. Größtmögliche ID-Übereinstimmung entscheidet
4. **Neighbor-Map** in Level unterteilt
5. Für statische Netze

Plaxton Schema

Die Grundlage

1. Knoten bekommt **node-ID** fester Länge (zB. B4F8)
2. Knoten enthält einen Wurzelblock
3. Größtmögliche ID-Übereinstimmung entscheidet
4. **Neighbor-Map** in Level unterteilt
5. Für statische Netze

Plaxton Schema

Die Grundlage

1. Knoten bekommt **node-ID** fester Länge (zB. B4F8)
2. Knoten enthält einen Wurzelblock
3. Größtmögliche ID-Übereinstimmung entscheidet
4. **Neighbor-Map** in Level unterteilt
5. Für statische Netze

Plaxton Schema

Beispiel

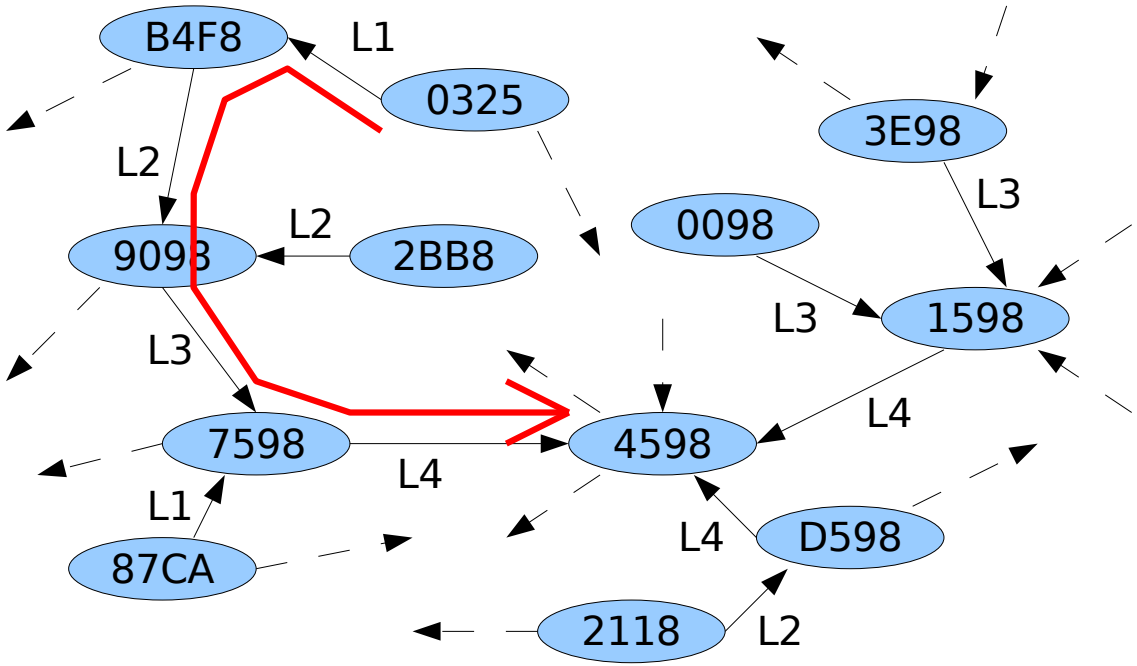

Abbildung: Das Plaxton "Routing- Meshäus [2]

Tapestry

Routingschicht

1. Erweiterung des Plaxton-Schemas
2. Gleiches Konzept der Namensvergabe
3. Fehlertolerantes Routing (unzuverlässige Knoten)
4. Für dynamische Netze (Einfügen von Knoten möglich)

Tapestry

Routingschicht

1. Erweiterung des Plaxton-Schemas
2. Gleiches Konzept der Namensvergabe
3. Fehlertolerantes Routing (unzuverlässige Knoten)
4. Für dynamische Netze (Einfügen von Knoten möglich)

Tapestry

Routingschicht

1. Erweiterung des Plaxton-Schemas
2. Gleiches Konzept der Namensvergabe
3. Fehlertolerantes Routing (unzuverlässige Knoten)
4. Für dynamische Netze (Einfügen von Knoten möglich)

Tapestry

Routingschicht

1. Erweiterung des Plaxton-Schemas
2. Gleiches Konzept der Namensvergabe
3. Fehlertolerantes Routing (unzuverlässige Knoten)
4. Für dynamische Netze (Einfügen von Knoten möglich)

Tapestry

Bekanntmachung von Objekten

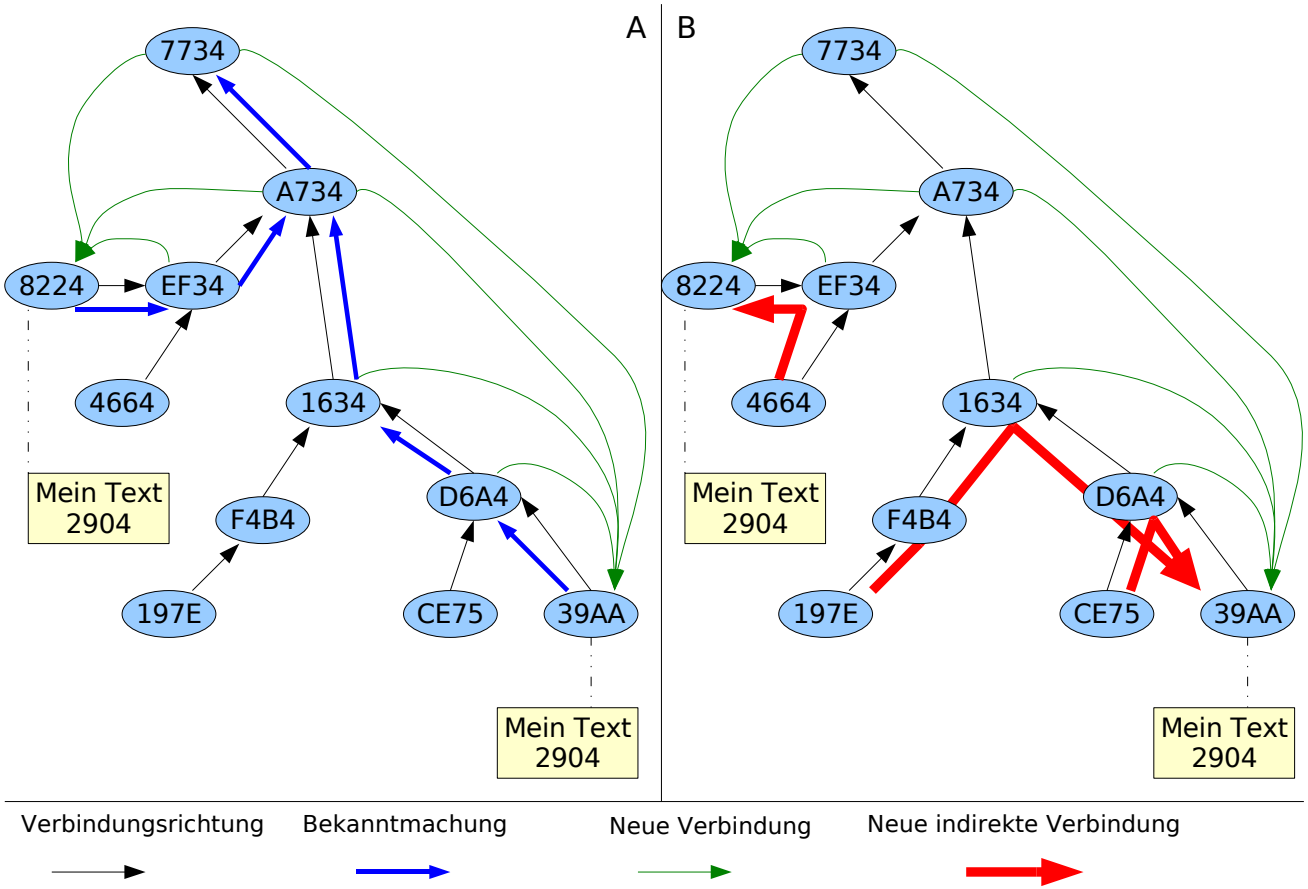

Abbildung: Bekannt machen von Objekten in Tapestry aus [3]

Implementierung

- Prototyp: **Pond**
  - Erste Implementierung in Java
- Noch nicht implementiert:
  - Introspection (Monitoring)
  - Zusammenspiel von „floating replicas“ und Archivierung
- OceanStore API:
  - Für UNIX
  - Als Browser-Plugin

Implementierung

- Prototyp: **Pond**
  - Erste Implementierung in Java
- Noch nicht implementiert:
  - Introspection (Monitoring)
  - Zusammenspiel von ,,floating replicas“ und Archivierung
- OceanStore API:
  - Für UNIX
  - Als Browser-Plugin

Implementierung

- Prototyp: **Pond**
  - Erste Implementierung in Java
- Noch nicht implementiert:
  - Introspection (Monitoring)
  - Zusammenspiel von ,,floating replicas“ und Archivierung
- OceanStore API:
  - Für UNIX
  - Als Browser-Plugin

Literaturverzeichnis

[1] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: The OceanStore prototype. In Proc. of USENIX File and Storage Technologies (FAST), 2003.

[2] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, und B. Zhao; OceanStore: An Architecture for Global-Scale Persistent Storage; In Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000); November 2000.

[3] S. C. Rhea und J. Kubiatowicz; Probabilistic Location and Routing; In Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), Juni 2002.

Ende

Vielen Dank für die Aufmerksamkeit!