Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung

Mykhaylo Kabalkin

Gemeinsamer Speicher in Collaborative Workspace

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Mykhaylo Kabalkin

Gemeinsamer Speicher in Collaborative Workspace

Ausarbeitung eingereicht im Rahmen der Vorlesung Anwendungen 1
im Studiengang Informatik Master of Science
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer : Prof. Dr. Kai von Luck

Abgegeben am 13. Juli 2006

Mykhaylo Kabalkin

Thema Ausarbeitung

Gemeinsamer Speicher in Collaborative Workspace

Stichworte

Collaborative Workspace, mobile Datenbanken, Google File System, BitTorrent, gemeinsamer Speicher

Kurzzusammenfassung

Gemeinsamer Speicher in Collabaorative Workspace ist heutzutage noch ein offenes Thema. In diesem Dokument beschreibt der Autor einige Technologien, auf die man bei dem Entwurf eines solchen Speichers aufbauen könnte, und setzt sein Ziel in dem Projekt „Collaborative Workspace“.

Mykhaylo Kabalkin

Title of the paper

Cooperative storage in Collaborative Workspace

Keywords

Collaborative Workspace, mobil data bases, Google File System, BitTorrent, cooperative storage

Abstract

Cooperative storage in collaborative workspace is nowadays still an open subject. In this paper the author describes some technologies, on which one could be based with the draft of such a storage, and sets his goal in the project "'Collaborative Workspace"'.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung und Motivation

Obwohl die Computer heutzutage eine große Rolle in unserem Leben spielen, gilt kollaborative Arbeit von Menschen als der wichtigste Bestandteil der Gesellschaft. Die Computer können den Leuten dabei helfen, aber sie werden sie meiner Meinung nach nie vollständig ersetzen.

Das Hauptziel, das im Rahmen der Vorlesungen Anwendungen 1 in dem zweiten Informatik Master-Semester sowie in Anwendungen 2, der Seminar-Ringvorlesung und dem Projekt in dem folgenden Semester verfolgt wird, ist es, einen Raum aufzubauen, der den Leuten kollaborative Arbeit erleichtern soll. Das Konzept des ähnlichen Raums wurde von der Forschungsgruppe HCI, unter der Leitung von Terry Winograd im Rahmen des Projekts IRoom [Johanson und Fox (2002)] vorgestellt. Den Aufbau des IRooms stellt die Abbildung 1.1 dar.

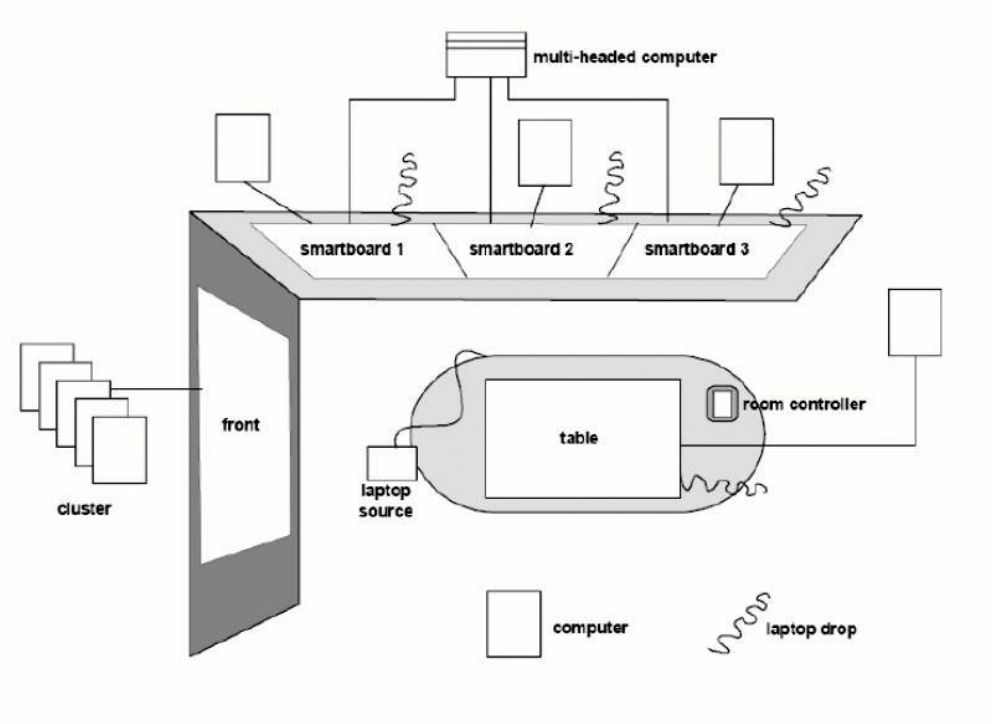

Abbildung 1.1: IRoom [Johanson und Fox (2002)], S.2

Im Laufe des nächsten Semesters wird sich der Autor dieser Ausarbeitung mit dem Zusammenhang von gemeinsamem Speicher und der Persistenz in Collaborative Workspace auseinander setzen. Die Ziele für das kommende Semester werden im Kapitel 3 beschrieben. Dieses Dokument beschreibt einige existierende Technologien, die verwendet oder auf die aufgebaut werden können, und wirft einen Blick auf Erkenntnisse und erste Ideen, die während der Vorlesung Anwendungen 1 gewonnen und diskutiert wurden.

2 Existierende Technologien

In einem Raum, in dem mehrere Menschen gemeinsam Dokumente ansehen und bearbeiten können, spielt der Begriff gemeinsamer Speicher eine große Rolle. Das System sollte auch eine Funktionalität zur Verfügung stellen, dass sich die Menschen um die Fragen des Speichers nicht kümmern müssen. D.h. die Benutzer dieses System haben keine Kenntnisse darüber, wo die Daten gespeichert werden, ob der Speicherplatz hierfür verfügbar ist und wie die Daten verfügbar werden. Die Technologien, die man in diesem Bereich verwenden kann, unterteilt der Autor in zwei große Bereiche, Mobile Datenbanken und File-basierte Systeme. Beide werden weiter unten detailliert beschrieben. Auch das Konzept von Lutz Behnke, das er in Rahmen des Masterprojekts „Ferienclub“ [UbiComp Projekte (2005/2006) erarbeitet hat und in seiner Masterarbeit weiter untersucht, wird von mir kurz angesprochen.

2.1 Mobile Datenbanken

Der zentrale Betrachtungspunkt eines Collaborative Workspaces ist der mobile Anwender, der sich, Informationen abgreifend und verwertend, in Raum und Zeit bewegt. Aus diesem Grund wird die mobile Datenbank Technologie hier als eine mögliche Lösung betrachtet und kurz beschrieben. Ob diese Technologie für unser Projekt in dem kommenden Semester eine passende ist, hängt davon ab, welche Anforderungen in diesem Projekt gesetzt werden, und für welche Clients das System entworfen wird.

Wenn die mobilen Clients Bestandteile des Gesamtsystems sein können, muss man den mobilen Kontext näher betrachten, der folgende Besonderheiten mit sich bringt:

- Unterschiedliche Bandbreiten zur Datenübertragung
- Keine ständige Netzwerkverbindung
- Keine permanente Anbindung an zentrale Datenbank
- Geringe Leistungsfähigkeit der mobilen Geräte
- Begrenzte Energiekapazität
- Beschränkte bzw. unterschiedliche Ein- und Ausgabemöglichkeiten

Weitere Problemfelder des mobilen Kontextes sind ebenfalls zu berücksichtigen:

- Datenverteilung
- Datenkonsistenz
- Ressourcennutzung (z.B. ortsabhängige Informationen)
- Powermanagement
- Benutzerschnittstellen

Mit den Einsatz von mobilen Datenbanken können oben beschriebe Probleme gelöst werden. Eins der wichtigen Ziele der mobilen Datenbanken ist die transparente Datenverteilung der Daten auf unterschiedliche Medien. Diese wird durch Fragmentierung und Allokation festgelegt, d.h. auf welchem Rechner wird ein Fragment der Daten gehalten, und welche Fragmente werden auf verschiedenen Rechnern repliziert gespeichert.

Durch den Einsatz der mobilen Datenbanken wird dem Benutzer erlaubt, eine Kopie des Datenbestandes auf sein Gerät zu replizieren, und wenn es nötig ist, außerhalb des Raums mit dieser Kopie zu arbeiten. Solche Funktionalität lässt sich durch Replikationsstrategien gewähren, die der Erhöhung der Effizienz, Ausfallsicherheit und Verfügbarkeit, und Autonomie dienen. Dadurch eine große Anzahl von Replikaten auf möglichst vielen Knoten steigt der Speicherplatzbedarf, aber der Datenverlust kann behoben werden.

Damit die Konsistenz der Daten innerhalb der Replikationsumgebung nach den lokalen Änderungen gesichert wird, müssen die lokalen Datenbestände mit dem globalen Datenbestand synchronisiert werden. Dies ist ein komplexer und aufwendiger Vorgang.

Wenn das System im kommenden Semester für mobile Clients entworfen werden sollte, wird der Autor zu der mobile Datenbank Technologie tendieren und das Konzept des gemeinsamen Speichers und der notwendigen Module auf dieser Technologie aufbauen. In diesem Fall können auch die gewonnenen Erkenntnisse in dem Masterprojekt „Ferienclub“ [UbiComp Projekte (2005/2006)] dabei als Grundlage dienen.

2.2 File-basierte Systeme

Außer der Verwendung der mobilen Datenbanken ist auch der Einsatz von file-basierten Lösungen sehr interessant. Einige Technologien werden hier beschrieben.

2.2.1 Google File System

Das Google File System (GFS) [GFS (2003)] wurde von Google Labs entworfen, um die schnell wachsende Anfrage von Google's Datenverarbeitungsprozessen zu verbessern. GFS ist ein Vertreter der Cluster-Filesysteme mit folgenden Zielen:

- hohe Performanz
- Skalierbarkeit
- Zuverlässigkeit
- Verfügbarkeit

Als weitere Motivation der Google-Entwickler war die Idee, dass die Fehler-Toleranz und das Auto-Recovery in das System integriert werden sollen.

Ein GFS Cluster besteht aus einem Master und mehreren Chunkservern und ist für eine Vielzahl von Clients zugreifbar. Jeder dieser Server ist in der Regel ein Linux Server. Der strukturelle Aufbau eines Clusters ist in der Abbildung 2.1 dargestellt.

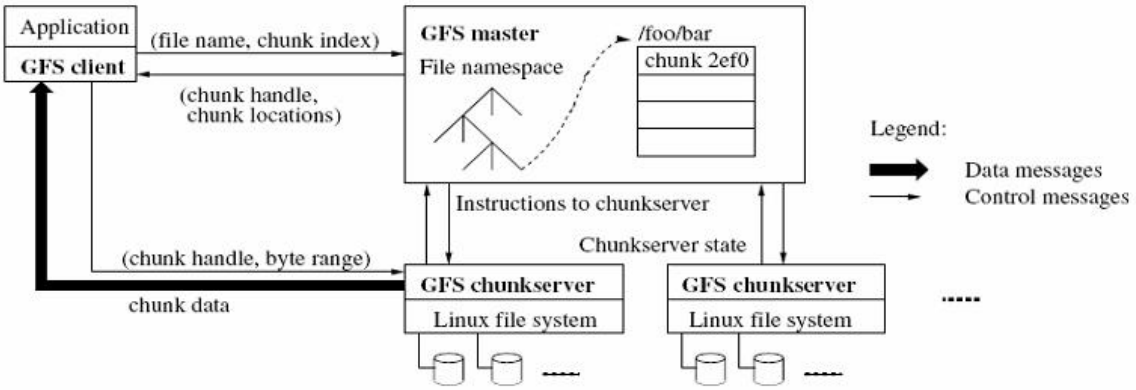

Abbildung 2.1: Google File System Architektur [GFS (2003), S.3]

Die Dateien werden in Chunks fester Größe unterteilt. Jeder Chunk ist mit einem unveränderlichen, globalen und einzigartigen 64 bit chunk handle identifiziert, der beim Erzeugen von dem GFS-Master zugewiesen wird. Chunkserver speichern diese auf die lokale Platte als Linux-Dateien. Für die Zuverlässigkeit wird jeder Chunk auf multiple Chunkserver repliziert. Als default sind drei Kopien von den GFS-Entwicklern vorgesehen. Lesend und schreibend werden Clients nie über den Master zugreifen, sondern ein Client fragt den Master, mit welchem Chunkserver er sich verbinden soll. Weiterhin werden Systemaktivitäten wie Chunk-Lease-Management, Garbage Collection und Chunk Migration zwischen den Chunkservern vom GFS-Master koordiniert.

Für das Google File System, als ein Beispiel für ein Cluster-Filesystem, sind der Umgang mit riesengroßen Datenmengen und die intelligente Ausbreitung der Arbeitslast sehr bedeutsame Aspekte. Dies wird dadurch erreicht, dass die eigentlichen Nutzdaten von den Metadaten getrennt gespeichert und verwaltet werden. Bei solcher Trennung können mehrere Speicherorte eingesetzt werden, ohne die Integrität des Gesamtsystems zu gefährden. Der GFS-Master hält alle Metadaten des Dateisystems. Diese beinhalten Datei- und Chunk-Namensräume, Zugriffsrechte sowie das Mapping von Dateien nach Chunks und die Position jeder Chunkkopie.

Da der GFS-Master nur koordiniert, hat er ein geringeres Datenvolumen als das Gesamtsystem zu handhaben. Die Replikation des GFS-Masters, als eine Redundanz der zentralen Komponente bei einem möglichen Ausfall, ist dadurch auch möglich.

Jede Mutation verändert die Chunkdaten und wird auf allen Chunk-Kopien durchgeführt. Leases (Kontrollrechte) werden hierbei benutzt, um eine konsistente Mutationsreihenfolge zu garantieren. Der GFS-Master wählt eine primäre Kopie aus, und gibt ihr das chunk lease, also das Recht der Wahl der Mutationsreihenfolge auf allen untergeordneten Chunk-Kopien. Die globale Mutationsreihenfolge ist somit durch die Leaserechte des Masters sowie durch die von der primären Kopie festgelegten Mutationsreihenfolge unter den anderen Kopien definiert.

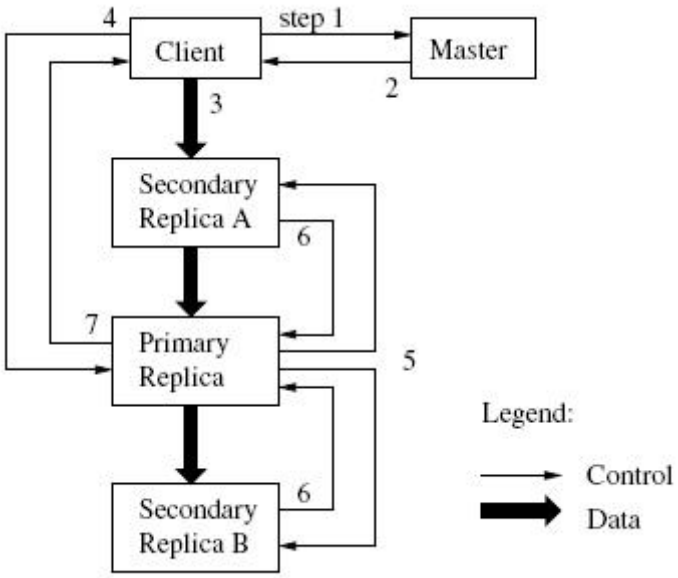

Abbildung 2.2: Schreibkontrolle und Datenfluss in GFS [GFS (2003), S.5]

Die Abbildung 2.2 stellt den Ablauf einer Schreiboperation in dem Google File System dar. Der Ablauf wird nach [GFS (2003)] beschrieben.

1. Der Client stellt zunächst eine Anfrage an den Master, welcher Chunkserver das Lease für den zu bearbeitenden Chunk hält. Existiert ein Lease, werden ebenfalls die Orte der übrigen Kopien zurückgeliefert. Kann kein Lease gefunden werden, so gewährt der GFS-Master einer Kopie ein Lease.
2. Der Master antwortet mit dem Ort der primären und aller sekundären Kopien. Der Client speichert diese Daten und muss zukünftig mit dem GFS-Master nur kommunizieren, falls die primäre Kopie nicht mehr erreichbar ist.
3. Der Client übermittelt die Daten an alle Kopien. Jeder Chunkserver speichert die gelieferten Daten in einem Buffer, bis die Daten gebraucht werden. Durch die Lösung des Datenflusses vom Kontrollfluss kann eine hohe Performance des Datenflusses garantiert werden.
4. Wenn alle Kopien die Daten empfangen haben, sendet der Client einen Schreibbefehl an die primäre Kopie. Der Schreibbefehl identifiziert die vorher gesendeten Daten und wird mit einer eindeutigen seriellen Nummer durch die primäre Kopie versehen.
5. Die primäre Kopie sendet den Schreibbefehl an alle sekundären Kopien weiter, so dass alle Kopien dieselben Mutationen mit denselben eindeutigen Nummern halten.
6. Die sekundären Kopien geben die Bestätigung der erfolgreichen Ausführung an die primäre Kopie zurück.
7. Die primäre Kopie antwortet dem Client mit dem Erfolg des Schreibbefehls oder eventuell aufgetretener Fehler. Im Falle von Fehlern wird der Client die Schritte 3 bis 7 wiederholen, bis der Schreibbefehl erfolgreich ausgeführt wurde. Gegebenenfalls muss der ganze Schreibbefehl neu ausgeführt werden. Für den Zeitraum der Schreiboperation sind die zu beschreibenden Dateiregionen für alle Clientzugriffe gesperrt. Nach dem erfolgreichen Schreiben werden sie wieder freigegeben.

Wenn eine Applikation eine große Datenmenge schreibt, wird der große Schreibbefehl in multiple kleinere Schreibbefehle unterteilt. Alle Schreibbefehle folgen dem oben beschriebenen Schema. Da mehrere Clients mehrere Schreiboperationen durchführen können, können die Daten durch konkurrierende Clients überschrieben werden.

2.2.2 BitTorrent

Als weitere mögliche Technologie sieht der Autor Peer2Peer Netze, insbesondere das BitTorrent kollaboratives Filesharing-Protokoll [Cohen (2003)], das unter Einsatz der File-Distribution-Technik für Tauschbörse gedacht ist.

Die Grundarchitektur von BitTorrent basiert auf dem Hybrid-P2P-Netzwerk, wobei der Server nicht im traditionellen Sinne arbeitet. BitTorrent hat keinen zentralen Server. Stattdessen finden sich die einzelnen Benutzer über einen so genannten Tracker zum Downloaden einer Datei zusammen und laden untereinander hoch. Der Tracker teilt jedem Peer (Client) die Adressen aller anderen Peers mit und sammelt gleichzeitig Informationen darüber, welche Peers welche Datei-Teile besitzen. In der Praxis bekommt ein Peer vom Tracker eine Zufallsliste mit den Peers, mit denen er Verbindung aufbauen kann. Der Tracker koordiniert die Up- und Download-Aktionen aller Peers. Die Abbildung 2.3 veranschaulicht die BitTorrent Architektur.

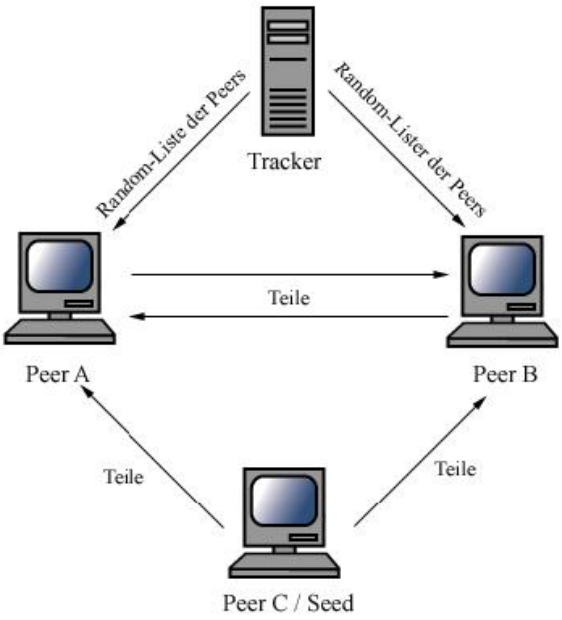

Abbildung 2.3: BitTorrent Architektur

Die Peers unterscheidet man in zwei Kategorien:

- **Seed** ist ein Peer, der die vollständige Datei besitzt und diese anderen Peers zum Download bereitstellt. Ein solcher Peer ist entweder der ursprüngliche Anbieter dieser Datei oder der, der nach einem kompletten Download einer Datei diese weiter zum Download zur Verfügung stellt.
- **Leecher** ist ein Peer, der die noch nicht komplette Datei heruntergeladen hat.

In neuen Versionen wurden „trackerlose“ Systeme entwickelt. Die Trackerfunktion wird dabei von den Clients mit übernommen. Dadurch wird u.a. die bisherig fehlende Ausfallsicherheit des Trackers vermieden. Der Einsatz von Tracker ist weiterhin möglich. Neue Systeme erleichtern auch das Anbieten der Dateien, da der Tracker den aufwendigsten Teil in dem

BitTorrent System ist. In diesem Fall kann der Tracker dezentral, als Distributed Hash Table [Cates (2003)] auf den Clients (im Netz) selbst abgelegt und verwaltet werden. Distributed Hash Table kann den Tracker vollständig ersetzen.

Eine der wichtigsten Komponenten des BitTorrent Systems ist die *.torrent*-Datei. Solche Dateien sind sehr klein (ungefähr zehn kByte), aber enthalten alle wichtige Metainformationen über die eigentlichen Nutzdaten (Dateien), die heruntergeladen werden sollen. Die Metadaten beinhalten folgende Informationen: Dateiname, Dateigröße, die Hashinformation und die Adresse des Track-Servers.

Clients können das BitTorrent System zu jedem beliebigen Zeitpunkt betreten und verlassen, ohne das das Gesamtsystem gefährdet wird. Dies ist ein sehr großer Vorteil, wenn man an den BitTorrent Einsatz in einem kollaborativen Raum denkt. Ein weiterer Vorteil kommt aus den Peer2Peer Netzwerken, dass es keine feste Bindung zwischen einem Objekt und seinem Speicherort gibt. Es existieren keine zentralen Einheiten und keine Hierarchie unter den Knoten (Clients).

Als Nachteil muss man berücksichtigen, dass keine Schreiboperation vorgesehen ist. Wenn ein Client eine Datei zum Download bereit stellt, wird sie aus Sicht der anderen Clients read-only. Nur nach dem vollständigen Download der Datei können andere Client sie verändern. Dadurch entsteht natürlich eine neue Datei. Das Löschen eines Objektes wird auch nur durch seinem Besitzer verwaltet, andere Clients haben keinen Einfluss darauf. Wenn ein Client sicher stellen will, dass ein Objekt zu einem späteren Zeitpunkt noch verfügbar ist, muss er es herunterladen und lokal speichern. BitTorrent stellt leider keine Suchfunktionalität zur Verfügung.

BitTorrent ist derzeit eine der besten Möglichkeiten, große Mengen legaler Daten schnell zu verbreiten, und ist sehr gut skaliert.

2.3 Kooperativer Speicher im Projekt “Ferienclub“

Lutz Behnke erarbeitete im Rahmen des Masterprojekts „Ferienclub“ [UbiComp Projekte (2005/2006)] ein Konzept [Behnke (2005)] und [Behnke (2006)], das sich um die Frage des kooperativen Speichers kümmert. Dieses Konzept wird von ihm in seiner Masterarbeit weiter untersucht.

Lutz Behnke hatte in seiner Arbeit folgende Ziele: Er wolle eine Schnittstelle bereit stellen, mit der sich Objekte ablegen, bei Bedarf wieder laden und mit Hilfe von Metadaten suchen lassen. Dabei würden Details wie Speicherort oder ähnliches vom Computer gehandhabt. Auch die Frage, ob ein lokaler Speicher langsam überliefe und bei nächster Gelegenheit alle Objekte, auf die schon lange nicht mehr zugegriffen würde, in einen anderen Datenspeicher

verschoben werden sollten, müssten nicht vom Benutzer beantwortet werden. Das gesamte System sollte mit einem Berechtigungs- und Sicherheitsmodell ausgestattet werden.

In einem informativen Gespräch brachte Lutz Behnke sein Konzept dem Autor dieser Ausarbeitung näher, der währenddessen einen großen Nachteil im Zusammenhang mit Collaborative Workspace feststellte. Das System von Lutz Behnke kümmert sich nur um die Frage des Speichers, d.h. sollte ein Benutzer eine Datei speichern wollen, kann er einfach z.B. „Save“ ausführen, ohne sagen zu müssen, wo diese Datei gespeichert werden soll. Danach kann diese von dem Eigentümer dieser Datei ohne Probleme zugegriffen werden. Aber in dem Fall, in dem auf diese Datei von anderen Benutzern zugegriffen werden könnte, ist es bis heute nicht vorgesehen. Dies widerspricht meiner Vorstellung von Collaborative Workspace, in dem die Benutzer gemeinsam die Daten bearbeiten können.

Obwohl die Ideen von Lutz Behnke sehr interessant sind, kann das Konzept aus oben genannten Gründen nicht im Rahmen des „Collaborative Workspace“ Projektes eingesetzt werden. Einige Ideen und Technologien kann man aber gebrauchen.

3 Ideen und Ziele

Für das Projekt in dem kommenden Semester setzte der Autor zusammen mit Philipp Roßberger und Pascal Pein folgendes Ziel:

- Das System, das im Rahmen des Projektes entworfen werden soll, soll auf der kollaborativen Arbeit mit Bildern basieren. Dabei ist ein mögliches Szenario vorgesehen wie z.B. der Entwurf eines Plakats / einer Collage mit Bildern.

Das Projekt wurde von uns zunächst in drei Bereiche unterteilt. In jedem Bereich werden einzelne Applikationen/Dienste entworfen, die unabhängig voneinander lauffähig sein, aber letztendlich aufeinander aufbauen sollen.

Eine Oberfläche für das Betrachten und/oder Manipulieren der Bilder wird von Philipp Roßberger entwickelt. Mit der Frage der Ähnlichkeit der Bilder wird sich Pascal Pein beschäftigen und einen Dienst mit einer Schnittstelle, z.B. über Webservices, zur Verfügung stellen, der das Anfordern einer Liste mit ähnlichen Bildern für ein bestimmtes Bild ermöglicht.

Der Autor dieses Dokumentes wird sich mit der Entwicklung eines Dienstes auseinander setzen, der das dynamische Verbinden von mehreren verteilten Speicherorten zu einem virtuellen gemeinschaftlichen Speicherort zulässt. Dabei gelten die im Kapitel 2 beschriebenen Technologien als Grundlagen für das Projekt im kommenden Semester.

Samba Share (http://samba.sernet.de/) wurde vom Prof. Dr. Kai von Luck als erste und schnelle Lösung vorgeschlagen. Ein Vorteil bei dem Einsatz von Samba ist es, dass sich das Speichermodul so verhält, als ob es schon eine fertige Komponente des gesamten Systems wäre.

Literaturverzeichnis

[Barroso, Dean und Hölzle 2003] BARROSO, Luiz André ; DEAN, Jeffrey ; HÖLZLE, Urs: *Web Search for a planet: The Google cluster architecture*. 2003. – URL http://labs.google.com/papers/googlecluster-ieee.pdf

[Behnke 2005] BEHNKE, Lutz: *Netzwerk-Transparente Persistenz*. 2005. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2005/behnke/abstract.pdf

[Behnke 2006] BEHNKE, Lutz: *Kooperativer Speicher: Schwächen und Gegenmaßnahmen*. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master05-06/behnke/abstract.pdf

[Cates 2003] CATES, Josh: *Robust and Efficient Data Management for a Distributed Hash Table*. 2003. – URL http://pdos.csail.mit.edu/papers/chord:cates-meng.pdf

[Cohen 2003] COHEN, Bram: *Incentives Build Robustness in BitTorrent*. 2003. – URL http://www.bittorrent.com/bittorrentecon.pdf

[Johanson und Fox 2002] JOHANSON, Brad ; FOX, Armando: *The Event Heap: A Coordination Infrastructure for Interactive Workspaces*. 2002. – URL http://graphics.stanford.edu/papers/eheap3/eheap_wmcsa.pdf

[GFS 2003] GHEMAWAT, Sanjay ; GOBIOFF, Howard ; LEUNG, Shun-Tak: *The Google File System*. 2003. – URL http://labs.google.com/papers/gfs-sosp2003.pdf

[Thomé 2005] THOMÉ, Mark: *Mobile Datenbanksysteme*. 2005. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2005/thome/abstract.pdf

[Thomé 2006] THOMÉ, Mark: *Mobile Informationssysteme für ortsbezogene Dienste*. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master05-06/thome/abstract.pdf

[UbiComp Projekte 2005/2006] FACHBEREICH INFORMATIK, HAW-Hamburg: *Masterprojekt "Ferienclub"*. 2005/2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projects.html