Fachhochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Seminararbeit: Zwischenspeicherung und Datenkonsistenz dynamischer Internet-Angebote unter Berücksichtigung hoher Seitenzugriffszahlen

5. Januar 2003

Stefan M. Dellbrügge
Fachbereich Angewandte Informatik
Fachhochschule Bonn-Rhein-Sieg
stefan.dellbruegge@inf.fh-bonn-rhein-sieg.de

Wintersemester 2002 / 2003
Verteilte und Parallele Systeme II
Prof. Dr. Rudolf Berrendorf

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Technische Infrastruktur des Untersuchungsobjektes

Die in dieser Ausarbeitung getroffenen Aussagen beziehen sich auf eine Untersuchung des *Thomas J. Watson Research Center* der IBM [Iyengar01]. Im Rahmen der Untersuchung wurden die Möglichkeiten geprüft, welche Beschleunigung im Lesezugriff auf eine Internet-Seite durch serverbasierte Cachekonsistenzprotokolle erreicht werden kann.

Das Datenmaterial wurde während der Olympischen Spiele 1998 in Nagano gesammelt. Das eingesetzte Serversystem bestand aus einem geographisch verteilten Mehrprozessorsystem der IBM RS/6000-Serie [IBM97]. Drei dieser Systeme waren in den Vereinigten Staaten an drei verschiedenen Standorten (Schaumburg, Columbus, Bethesda) installiert. Ein weiteres System wurde in Tokio installiert. Insgesamt enthielt dieses System 120 Einzelprozessorknoten. Durch sogenannte *Network Dispatcher* Router (ND) werden eingehende Anfragen an weniger belastete Knoten verteilt. Das angeforderte Dokument wird direkt an den Client versendet, ohne nochmals den Network Dispatcher zu passieren.

Mit der beschriebenen Infrastruktur repräsentiert das System eine durch die zunehmende Kommerzialisierung wachsende Klasse von Internet-Seiten. Informationen über aktuelle Sportgroßveranstaltungen, Nachrichten oder E-Commerce-Anwendungen (Shopsysteme, Aktienhandel) werden im Internet einer beliebigen Anzahl von Konsumenten bereitgestellt. Hierzu gehören zum Beispiel die Internet-Seiten der Tennismeisterschaften in Wimbledon, der Nachrichtenagenturen (reuters, CNN) und großer Finanzdienstleister (Schwab).

1.2 Analyse der Zugriffmuster von dynamischen Internetseiten

Ein gemeinsames Merkmal dieser Internet-Dienste ist die Dynamik der Hintergrunddaten, aus denen die Internet-Seiten generiert werden. Die Datenveränderungen erfolgen dabei nicht deterministisch. In Kombination mit einer kurzen Zeitspanne (Einheit: Minuten) zwischen den Änderungen ist ein Zwischenspeichern solcher Seiten nicht, oder nur sehr schwer möglich. Weiteres Merkmal ist die Zugriffsanzahl pro Tag oder die Anzahl der Zugriffe pro Minute. Die Internet-Seiten der Olympischen Spiele von 1998 verzeichneten an einem Tag rund 56,8 Millionen Zugriffe. Das entspricht über 40.000 Zugriffen pro Minute. Dieser Wert beschreibt eine Grundlast für das System, der in sogenannten *Peaks* überschritten wird. Neben den genannten 56 Millionen Zugriffen pro Tag wurden von dem beschriebenen System in einer Minute als Maximum über 110.000 Zugriffe bearbeitet.
Aus den genannten Zahlen läßt sich die Bandbreite der Leistungsanforderungen an ein hochdynamisches Internetangebot ablesen. Verbindet man die von Extern gestellten Anforderungen bezüglich der Zugriffsanzahl mit den immanenten Abhängigkeiten zwischen Daten und Internet-Seiten, dann lassen sich die Anforderungen an ein Konsistenzprotokoll für ein vergleichbares Internetangebot wie folgt beschreiben:

- schnelle Reaktion auf Änderungen der Hintergrunddaten
- festgelegtes maximales „Alter“ eines Objektes im Cache
- garantierte Aktualität der Cache-Objekte

	Objekte		Anfragen	
	Anzahl	%	Anzahl	%
Bilder	10 365	17.1	27 704 815	48.7
HTML	8 702	14.4	4 546 990	8.0
dynamisch	36 942	61.0	16 921 760	29.8
anderes	4 528	7.5	7 657 369	13.5
Total	60 537	100	56 830 934	100

Tabelle 1: Objekte der Internet-Seiten der Olympischen Spiele 1998 [Iyengar99]

- geringer Protokoll-Overhead beim Abgleich zahlreicher Caches

Aus einer Analyse der Zugriffsprotokolle der Internet-Seiten der Olympischen Spiele von 1998 wurde deutlich, daß die genannten Forderungen an das Konsistenzprotokoll nicht für alle Cache-Objekte zutreffen. Zugriffe lassen sich danach unterscheiden, ob dynamische Seiten angefordert wurden oder statische Objekte (Grafiken, Audio, Applets). Es ist offensichtlich, daß nur dynamische Seiten von den häufigen Änderungen der Hintergrunddaten betroffen sind. Statische Objekte verändern sich verglichen dazu in größeren Zeitabständen. Ergebnis der Analyse war, daß ungefähr 30% aller Anfragen dynamische Inhalte betreffen (vgl. Tabelle 1). D.h. alle Optimierungen des Cachekonsistenzprotokolls wirken insbesondere auf diese Objekte.

1.3 Vorhandene Datenkonsistenzprotokolle im Internet

Die aktuelle Version des Hypertext Transfer Protocol (HTTP 1.1) verwendet ein einfaches Modell zur Erhaltung der Cache-Konsistenz. Basis dieses Modells ist das sogenannte *client polling*. Der HTTP-Client erfragt dabei bei einem Server die Gültigkeit eines im Cache gespeicherten Objektes. In den verfügbaren Implementierungen dieses Konsistenzprotokolls tritt häufig der Fall ein, daß ein Client vor der Auslieferung des Objektes aus dem Cache den Server nach dessen Gültigkeit befragt. Ist das Objekt gültig, also noch nicht veraltet, dann bestätigt der Server die Gültigkeit.

Dieses Konsistenzprotokoll ist in HTTP 1.1 mit dem Parameter *Time to live* (TTL) implementiert. Damit wird jedem Objekt im Cache eine Zeitspanne zugeordnet, in der dieses Objekt vom Cache als noch gültig angesehen wird. Ist diese Zeitspanne überschritten und es soll erneut auf dieses Objekt zugegriffen werden, muß der Client das Objekt revalidieren. Dazu enthält HTTP eine `Get-if-modified-since`-Anfrage. Der Server kann dann mit der neuen Version des Objektes antworten („`200 OK`“) oder er bestätigt die Gültigkeit des im Cache gespeicherten Objektes („`304 not modified`“). Wie aus dem beschriebenen Ablauf deutlich wird, ist das verwendete Konsistenzprotokoll in den folgenden Punkten eingeschränkt:

- jedes Cache-Objekt besitzt eine individuelle Gültigkeit (TTL),
- der *Time to live*-Parameter unterscheidet nicht zwischen durchschnittlichem und maximal möglichem Alter eines Cache-Objekts.

Diese Charakteristika des HTTP-Cachenkonsistenzprotokoll begrenzen die Leistungsfähigkeit dieser Implementierung. Eine Begrenzung entsteht durch den objekt-bezogenen

TTL-Parameter. Läuft im Client-Cache die Gültigkeit einer Menge von Objekten zur gleichen Zeit ab, dann muß bei weiteren Zugriffen für jedes ungültige Objekt der TTL-Parameter individuell neu gesetzt werden. Die dadurch notwendigen Serveranfragen und -antworten erhöhen die Last des Servers und verzögern den Cachezugriff.

Die fehlende Unterscheidung zwischen durchschnittlichem und maximal möglichem Alter eines Cache-Objektes verhindert die Definition und Implementierung von speziellen Dienstmerkmalen. Ein mögliches Dienstmerkmal ist zum Beispiel im Falle eines Online-Nachrichtendienstes die Garantie, keine Tickerdaten auszuliefern, die älter als fünf Minuten sind. Dieses Merkmal betrifft das maximale Alter von Cache-Objekten. Darüberhinaus kann gewünscht sein, alle neuen Nachrichten innerhalb von wenigen Sekunden zu veröffentlichen. Für dieses Dienstmerkmal ist der Parameter des durchschnittlichen Alters von Cache-Objekten relevant.

2 Datenkonsistenz in hoch-dynamischen Internetangeboten

2.1 Funktionsweise

Die beschriebenen Leistungsbegrenzungen von Client Polling-Protokollen lassen sich mit einem *serverbasierten* Konsistenzprotokoll teilweise umgehen oder aufheben. Ein serverbasiertes Konsistenzprotokoll teilt den Cache-Clients Veränderungen der Original-Objekte selbstständig mit (Server-Push-Technik). Durch dieses Vorgehen können Caches Objekte an Clients weitergeben, ohne vorher beim Server die Gültigkeit des Objektes prüfen zu müssen. Wird ein Original-Objekt geändert, dann ist es Aufgabe des Servers, alle Cache-Clients darüber zu informieren. Im Zusammenhang mit den beschriebenen hoch-dynamischen Internet-Angeboten erscheint dieses Konsistenzprotokoll besonders sinnvoll zu sein:

- Client Polling-Protokolle müssen für möglichst aktuelle Cache-Objekte optimiert werden. Dazu ist eine niedrige Anfragefrequenz notwendig, die zu Protokolloverhead und unnötiger Serverlast führt. Serverbasierte Konsistenzprotokolle senken die Serverlaste und erlauben eine Lastverteilung durch den Server.
- Ein Caching dynamischer Objekte erhöht zusätzlich die Leistung des Internetangebots. Anfrage und Generierung dynamischer Inhalte benötigen deutlich mehr Zeit im Vergleich zu statischen Daten. Ein Caching senkt damit die Leseverzögerung für anfragende Clients.

Die folgende Betrachtung eines serverbasierten Konsistenzprotokolls benötigt zwei Parameter, die eine Beschreibung der Dienstgüte eines Protokolls zulassen. Diese beiden Parameter sind maximales Alter (engl. *worst case staleness*) und durschnittliches Alter (engl. *average staleness*) eines Cache-Objektes. Das maximale Alter eines Cache-Objektes wird mit $\Delta(t)$ beschrieben. Gilt $\Delta(t)$-Konsistenz, dann ist ein Cache-Objekt nie älter als t. D.h. wenn das letzte Update eines Cache-Ojektes zum Zeitpunkt T erfolgt ist, dann muß ein Zugriff zum Zeitpunkt $T + t$ eine neue Version des Objektes aus dem Cache liefern. Das durchschnittliche Alter eines Cache-Objektes bemisst sich an zwei Faktoren. Zum einen der Anteil der Lesezugriffe auf den Cache, die bereits ungültige Objekte liefern und die durchschnittliche Wartezeit in Sekunden, die vergangen sind, seit dem das Cache-Objekt ungültig geworden ist.

Konsistenzprotokolle setzen diese Parameter mit Hilfe von Fristen (engl. *leases*) um. Das maximale Alter eines Cache-Objektes beschreibt mit diesen Fristen den Zeitraum, den ein Cache-Client das Objekt zwischenspeichern darf ohne erneut mit dem Server zu kommunizieren (vgl. TTL in HTTP). Nicht enthalten ist in diesem Konsistenzmodell eine Betrachtung des durchschnittlichen Alters eine Cache-Objektes. Serverbasierte Konsistenzprotokolle trennen diese beiden Parameter durch das Prinzip der Volumenfristen (engl. *volume leases*). In diesem Modell werden objektbezogene Fristen festgelegt, sowie für eine Menge von zusammengehörigen Objekten eine gemeinsame Frist definiert.

Mit jedem angefragten Objekt enthält der Cache-Client eine objektbezogene Frist. Diese objektbezogene Frist wird vom Server gespeichert und bei einer Änderung des Original-Objektes erneut gesetzt. Der Server informiert dann alle Cache-Clients, die das betroffene Objekt zwischengespeichert haben, über die Änderung der Frist. Damit sind die Cache-Clients über die Änderung des Original-Objektes informiert und können diese Objekt neu beim Server anfordern.

Ein Protokoll mit Volumenfristen dehnt die Lebenszeit eines Cache-Objektes aus und bezieht andere Objekte in diese Frist ein. Dabei werden Objekte berücksichtig, die zusammenhängen. Dies sind zum Beispiel alle Grafiken einer Internet-Seite oder alle Ergebnisse einer Datenbankabfrage. Durch den Bezug zu anderen Objekte kann eine Volumenfrist länger sein, als die objektbezogene TTL-Frist. Sobald aber die Volumenfrist abgelaufen ist, wird auch das bereits abgelaufene Einzel-Objekt revalidiert.

2.2 Synchronisation

Die Synchronisation der Datenkonsistenz zwischen Server und Cache-Client erfolgt durch die Anwendung des Konsistenzprotokolls. Es bleibt zu definieren, wie das Protokoll mit Fehlerzuständen umgeht und eine Resynchronisation ermöglicht. Eine Resynchronisation ist dann nötig, wenn zum Beispiel der Client keine Verbindung zum Server herstellen konnte (oder umgekehrt) und sich einzelne Datenobjekte im Server verändert haben. Allgemein können die Fehlerzustände in zwei Gruppen unterschieden werden:

konsistenzerhaltend: Hierzu gehören Verbindungfehler aufgrund von Netzwerkstörungen. Es bleibt der Datenzustand erhalten, der vor Auftreten des Verbindungsfehlers vorlag.

konsistenzverletzend: In diese Gruppe gehören schwerwiegende Fehler wie z.B. Server- oder Clientausfälle. Nach einem solchen Fehler verfügt der Cache-Client über keine oder nur noch sehr wenige gültige Cacheobjekte.

Im folgenden werde drei Vorgehensweise erläutert, mit denen sich Client und Server resynchronisieren können. Der Schwerpunkt der Betrachtung liegt darauf, ein möglichst effektives Verfahren zu ermitteln, das skalierbar ist und den Server nicht über Gebühr auslastet.

Anforderungsabhängige Revalidierung: Die langsamste Strategie der Resynchronisation entspricht annähernd dem normalen Konsistenzprotokollablauf und wird als anforderungsabhängige Revalidierung (engl. *demand revalidation*) bezeichnet. Nach einem Fehlerzustand und dem erneuten Verbindungsaufbau werden alle Objekte im Cache des Clients als ungültig markiert. Dies geschieht selbstständig durch den Client. Anschließend wird jedes Objekt einzeln erneut validiert, wenn es aus dem Cache abgerufen wird.

Sofortige Revalidierung: Eine zügige und mit geringen Datenmengen effektive Strategie wird als sofortige Revalidierung (engl. *immediate revalidation*) bezeichnet. In diesem Verfahren werden nach dem Verbindungsaufbau alle im Cache des Clients gespeicherten Objekte sofort vollständig revalidiert. Wie angedeutet, ist diese Strategie nur mit kleinen Objektmengen effektiv nutzbar. Die sofortige Revalidierung aller Cache-Objekte kann den Zugriff des Clients auf den Server dadurch verzögern, daß erst große Datenmengen zwischen Server und Client ausgetauscht werden müssen, bevor der Cache ein gültiges Objekt zurückliefern kann. Um dieses Problem zu umgehen, kann die sofortige Revalidierung gleichzeitig zu den normalen Zugriffen des Clients auf den Server erfolgen (engl. *background revalidation*).

Es ist einsichtig, daß die Vorteile der sofortigen oder Hintergrund-Revalidierung nur greifen, wenn die Zeit der Verbindungsunterbrechung relativ kurz ist und nur wenige Cacheobjekte revalidiert werden müssen. Nach einer längeren Verbindungsunterbrechung ist es dagegen sinnvoll, mit anforderungsabhängiger Revalidierung den Cache zu synchronisieren. Diese Annahme gilt aber nur für Fehlerzustände, die konsistenzerhaltend sind. Ist durch einen konsistenzverletzenden Fehler der Cache ungültig, ist die Hintergrund-Revalidierung vorzuziehen, da hierbei schneller ein konsistenter Zustand erreicht werden kann.

Die Abbildungen 1 und 2 zeigen die erfolgreichen Cachezugriffe nach einem konsistenzerhaltenden Fehlerzustand von einer Sekunde (Abb. 1) sowie 1.000 Sekunden (Abb. 2) Dauer. Es wird deutlich, daß mit sofortiger Revalidierung schon kurze Zeit nach der erneuten Verbindung gute Trefferraten erzielt werden können. Dauert die Verbindungsunterbrechung länger, tritt eine Verschlechterung der Trefferrate bei sofortiger Revalidierung ein. Die anforderungsabhängige Revalidierung liefert in beiden Fehlerfällen gleichbleibende Ergebnisse, die grundsätzlich unter denen einer sofortigen Revalidierung liegen.

2.3 Leistungscharakteristik

Die in den vorhergehenden Abschnitten beschriebenen Verfahren zur serverbasierten Datenkonsistenz wurden in einer Simulation auf einen Auszug von rund 9 Millionen Datensätzen aus den Zugriffsprotokollen des Internet-Angebots der Olympischen Spiele von 1998 angewendet. Im Rahmen dieser Leistungsmessung wurde besonders die Verzögerung des Lesezugriffs berücksichtigt. Dabei ist anzumerken, daß der Lesezugriff in zwei Fällen besonders verlangsamt wird:

1. ein Objekt ist nicht im Cache vorhanden (engl. *cache miss*)
2. Die Gültigkeit eines Objektes muß bestätigt werden (engl. *consistency miss*)

Diese beiden Fälle sind in der weiteren Betrachtung nicht explizit ausgewiesen. D.h. die Simulationen geben den Gesamtcharakter des Cachesystems wieder. Eine detaillierte Betrachtung der Ursachen für einzelne Merkmale ist nicht erfolgt.

Die Abbildungen 3 und 4 stellen die Leistungsfähigkeit eines Volumen-Fristen basierten Konsistenzprotokolls im Vergleich zum verfügbaren TTL-Protokoll dar. Auf der x-Achse wird dabei das maximale Alter (*worst case staleness*) eines Cache-Objektes aufgetragen. Die y-Achse zeigt die damit erreichten erfolgreichen lokalen Cache-Zugriffe eines Clients. Abbildung 3 zeigt, daß ein Volumen-Fristen Protokoll bei einer deutlich kürzeren maximalen Frist (100 Sekunden), mehr erfolgreiche Treffer liefert. Ein TTL-Protokoll kann vergleichbare Trefferraten (ca. 40 %) nur bei

einer Frist ab 10.000 Sekunden garantieren. In diesem Zusammenhang ist darauf hinzuweisen, daß bei dieser Fristenlänge die Anzahl der falschen Zugriffe (Abb. 5) und das durchschnittliche Alter eines Cache-Objektes dramatisch ansteigen (Abb. 6). Dies verschlechtert die Effektivität des TTL-Protokolls gegenüber einer volumenbasierten Lösung, die mit einer niedrigeren Frist schon vergleichbare Ergebnisse liefert. Gleichzeitig garantiert ein Volumen-Fristen Protokoll einen aktuelleren Cacheinhalt und damit aktuellere Information für den Anwender (vgl. Dienstmerkmale eines Nachrichtendienstes, Seite 4).

Neben den allgemeinen Leistungsmerkmalen eines serverbasierten Konsistenzprotokolls ist das Verhalten bei der Zwischenspeicherung von dynamischen Internet-Inhalten besonders interessant.

Die Ergebnisse einer Simulation unter diesem Fokus zeigt Abbildung 7. Es ist erkennbar, daß ein Volumen-Protokoll bei kürzeren maximalen Fristen dem TTL-Protokoll überlegen ist. Für dynamische Inhalte werden mit Fristen bis 100 Sekunden vergleichbare positive Trefferquoten erzielt, wie für statische Inhalte. Erst ab einer Frist von rund 1000 Sekunden ist das TTL-Protokoll gleichwertig. Ab diesen Fristen gilt aber ebenfalls das oben gesagte bezüglich Cache-misses und durchschnittlichem Alter eines Cache-Objektes.

2.4 Skalierung

Die Skalierbarkeit eines serverbasierten Konsistenzprotokolls für Internet-Angebote der beschriebenen Leistungsklasse kann durch verschiedene Faktoren eingeschränkt sein:

- Mit zunehmender Zahl der Clients (c), steigt die Anzahl der gespeicherten objektbezogenen Fristen (o) im Server. Der Speicherbedarf ist proportional zur Anzahl der Clients und der zwischengespeicherten Objekte ($c \times o$). Eine Untersuchung des Internet-Angebots der Olympischen Spiele hat ergeben, daß die Speicherbelegung über einen Zeitraum von 24 Stunden linear steigt. Nach diesem Zeitraum wurden rund 5 Millionen Fristen pro Knoten gespeichert. Diese Fristen benötigten rund 295 MB Arbeitsspeicher, womit 0,4% der Gesamtspeicherkapazität des Systems belegt waren.
- Die Invalidierung häufig abgefragter Objekte erzeugt Lastspitzen durch den Versand der Invalidierungsnachricht an zahlreiche Clients. Die Anzahl der Nachrichten ist hierbei ebenfalls proportional zur Anzahl der Clients und der invalidierten Objekte je Client.

Für diese beiden Problemfelder müssen Lösungen gefunden werden, wenn ein serverbasiertes Konsistenzprotokoll effektiv eingesetzt werden soll. Bisher sind verschiedene Techniken hierzu überlegt worden, die im Folgenden kurz vorgestellt werden:

Cache-Hierarchie: In einer Cache-Hierarchie versendet der Server Invalidierungsnachrichten nur an die Caches der unmittelbar nächsten Ebene. Die nachfolgenden Ebenen werden von den Caches der 2. Ebene bedient.

„gezielter" Multicast: Mit dem „gezielten" Multicast soll erreicht werden, daß Client-Caches nur Invalidierungsnachrichten für die von ihnen gespeicherten Objekte erhalten. Möglich wird dies durch die Anwendung von Mulitcast-Kanälen auf einer objektbezogenen Basis oder durch Filterung der Multicast-Nachrichten.

Verzögerte Invalidierungsnachrichten: Invalidierungsnachrichten werden frühestens an einen Cache-Client gesendet, wenn dessen Volumenfrist abgelaufen ist. Solange eine gültige Volumenfrist für den Client vorhanden ist, werden alle Invalidierungsnachrichten gespeichert. Eine Abwandlung dieser Technik ist die *Hintergrund-Invalidierung*. Invalidierungsnachrichten werden nur versendet, wenn der Server hierfür Rechenkapazität und Netzwerkbandbreite zur Verfügung hat.

Leerlauf-Clients: In diesem Modell wird angenommen, daß ein Cache-Client, der nach einer vordefinierten Zeitspanne nicht auf eine Invalidierungsnachricht reagiert hat, nicht erreichbar oder mit anderen Aktivitäten beschäftigt ist. Der Server verwirft nach dieser Zeitspanne alle gespeicherten Fristen zu diesem Cache-Client und seinen Objekten und reduziert damit seine Speicherbelegung.

Die genannten Probleme eines serverbasierten Konsistenzprotokolls lassen sich mit diesen Techniken teilweise lösen oder reduzieren. Problematisch sind aber die Auswirkungen auf das Gesamtverhalten des Cache-Systems, insbesondere durch den Einsatz der *Leerlauf-Client*-Technik bzw. der *verzögerten Invalidierungsnachrichten*. Im ersten Fall muß bei einer zu kurzen Zeitspanne ein unnötiger Protokoll-Overhead durch einen erneuten Verbindungsaufbau zwischen Server und Cache-Client und die Resynchronisation in Kauf genommen werden. Damit steigt die Wahrscheinlichkeit von Konsistenzfehlern. Gleichzeitig steigt im Falle verzögerter Invaldierungsnachrichten das durchschnittliche Alter der Cache-Objekte. Damit erhöht sich die Anzahl der nicht validierten Objekte im Cache.

2.5 Prototypenentwicklung

Das beschriebene serverbasierte Konsistenzprotokoll mit Volumen-Fristen wurde in einem Squid-Webcache 2.2.5 als Prototyp implementiert und getestet. Der Testaufbau bestand aus zwei Clients, einem Server sowie einem vorgeschalteten Proxy.
Die ersten verfügbaren Testergebnisse konnten keinen Leistungsvorteil für ein serverbasiertes Konsistenzprotokoll nachweisen. Die Last des Konsistenz-Proxy stieg gegenüber einer normalen Squid-Installation um weniger als 3 Prozent. Zusätzlich verzögerte sich der Cache-Zugriff um rund 5 Prozent. Damit sind die durch die Simulationen der vorhergehenden Kapitel erwarteten Leistungssteigerungen eines serverbasierten Konsistenzprotokolls nicht erreicht worden.

3 Abschlussbetrachtung

Die in den vorhergehenden Abschnitten geschilderten Überlegungen zum Einsatz eines serverbasierten Konsistenzprotokolls bearbeiten die Problemstellung nur theoretisch. In der Implementierung des Prototypen konnten die ermittelten Leistungssteigerungen nicht nachgewiesen werden. Es ist dabei davon auszugehen, daß die Simulation Randbedingungen nicht ausreichend beachtet hat oder die Implementierung des Konsistenzprotokolls auf den vorhandenen Systemen durch nicht bekannte Faktoren beeinflußt wurde. Somit bleiben die genannten Vorteile eines serverbasierten Konsistenzprotokolls bisher nur theoretisch erfassbar.

Eine Implementierung eines serverbasierten Konsistenzprotokolls ist zur Zeit nicht verfügbar. Trotz der positiven Beurteilung der Skalierung des evaluierten Systems (vgl. 2.4) ist zu bedenken, welche Anzahl von Internet-Seiten dieser Leistungsklasse angehört und angehören wird. Die Bedeutung dieser Angebote wird steigen, aber

höchstwahrscheinlich nur einen Bruchteil der Gesamtzahl aller verfügbaren Internet-Seiten wird auf dieser Technologie basieren. Daher ist das Anwendungsgebiet einer solchen Implementierung eingeschränkt. Darüberhinaus ist eine Implementierung nur dann effektiv, wenn alle zugreifenden Clients ebenfalls das Protokoll implementieren. Aus diesem Grund ist es wahrscheinlich, daß die ersten Einsatzgebiete eines serverbasierten Konsistenzprotokolls eine Server-Proxy-Lösung umfassen. Das in der Untersuchung dargestellte Modell einer echten Server-Client-Implementierung ist aus den genannten Gründen unwahrscheinlich.

A Abbildungen

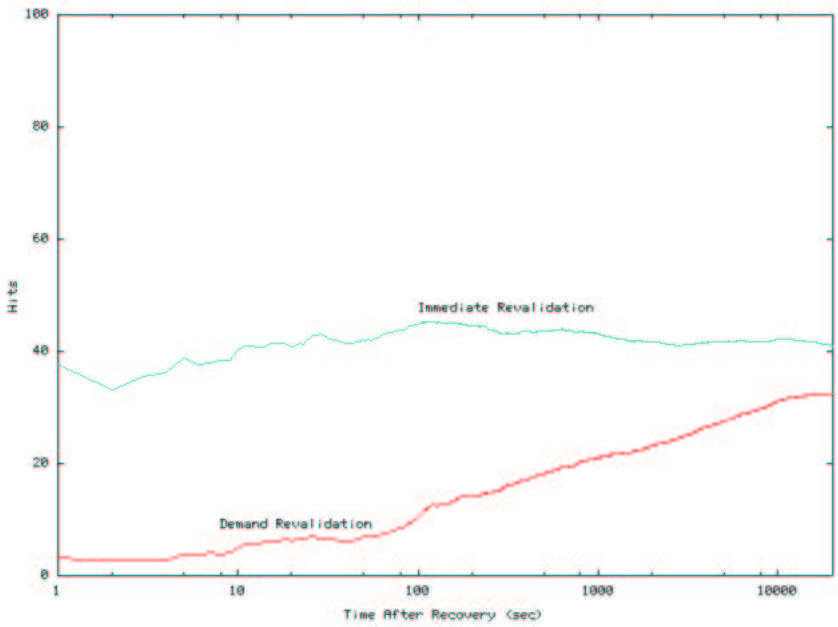

Abbildung 1: Erfolgreiche Cachezugriffe mit TTL-Fristen nach Unterbrechung von 1 Sekunde

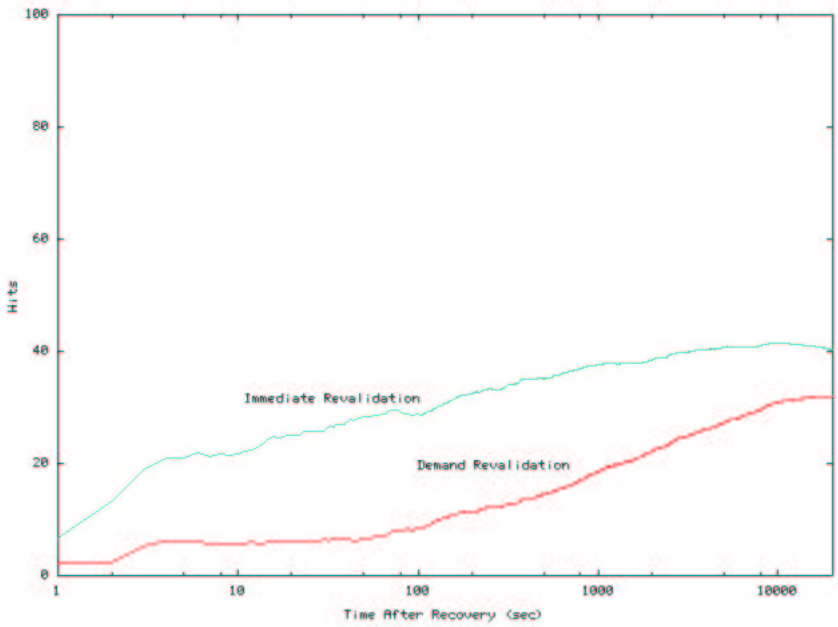

Abbildung 2: Erfolgreiche Cachezugriffe mit TTL-Fristen nach Unterbrechung von 1 000 Sekunden

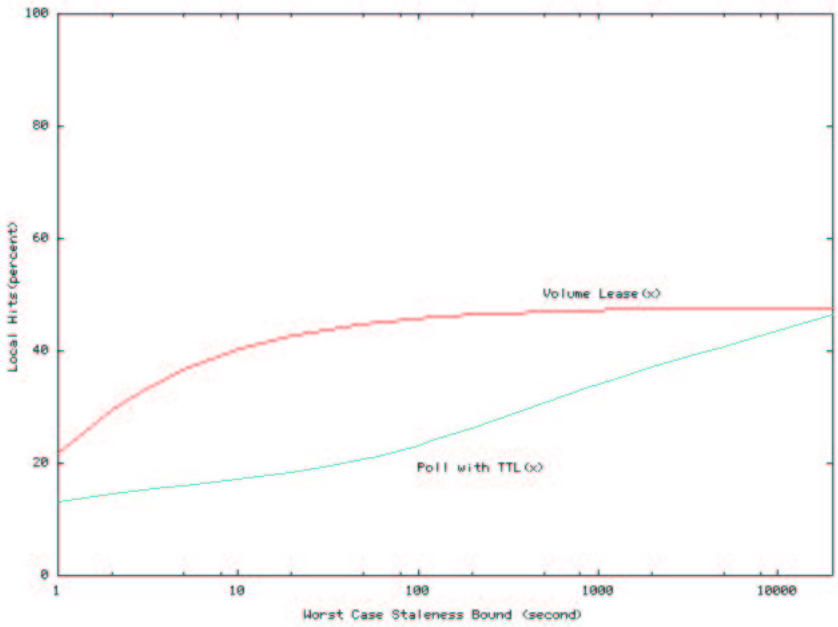

Abbildung 3: Erfolgreiche Cachezugriffe mit TTL- und Volumen-Fristen

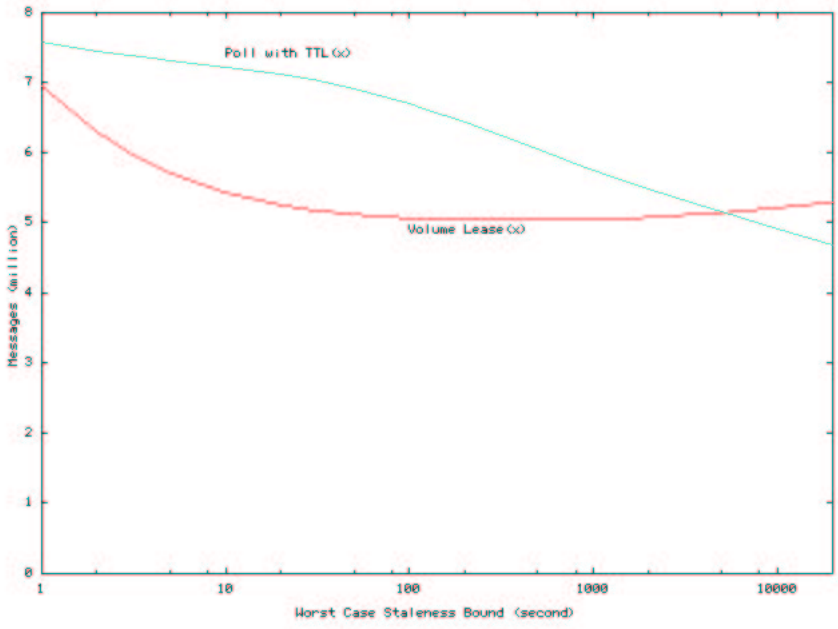

Abbildung 4: Invalidierungsnachrichten mit TTL- und Volumen-Fristen

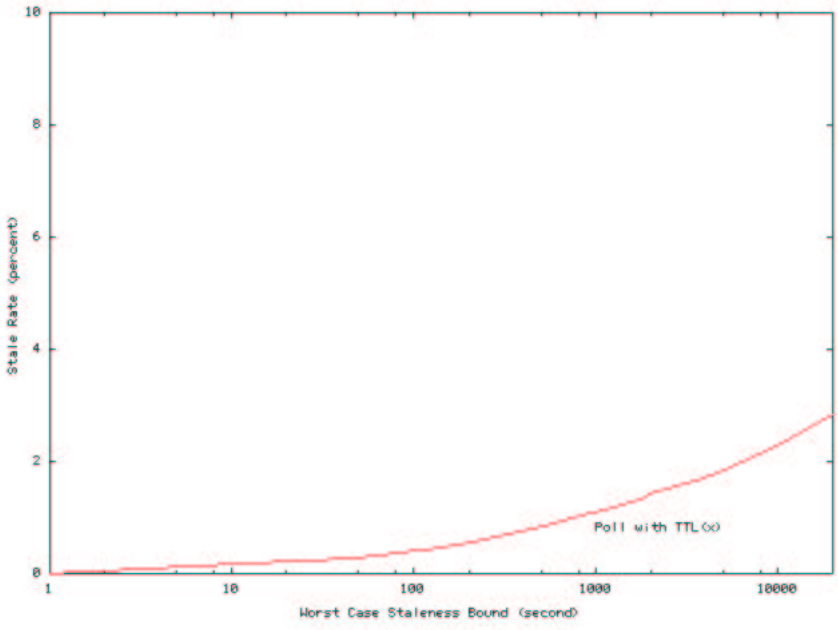

Abbildung 5: Cache Misses mit TTL-Protokoll

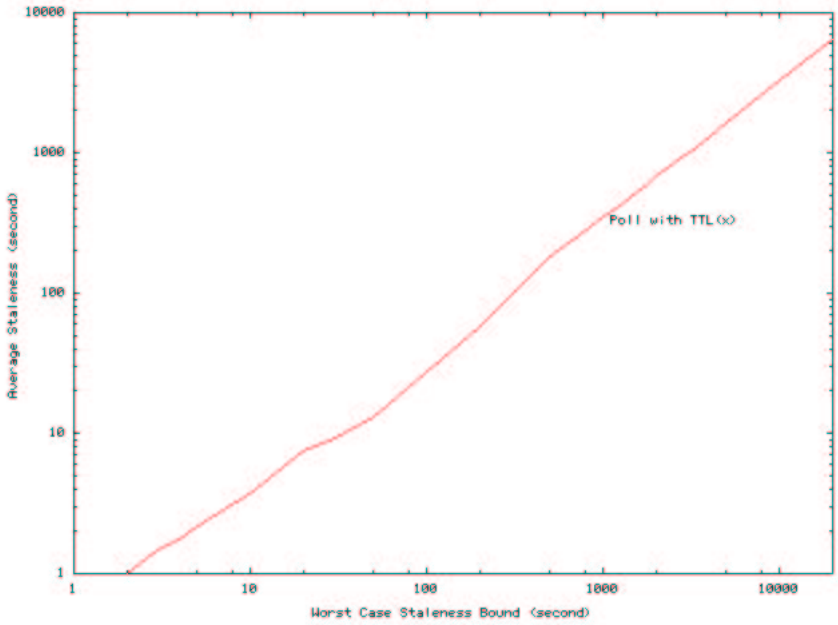

Abbildung 6: Durchschnittliches Alter eines Cache-Objektes mit TTL-Protokoll

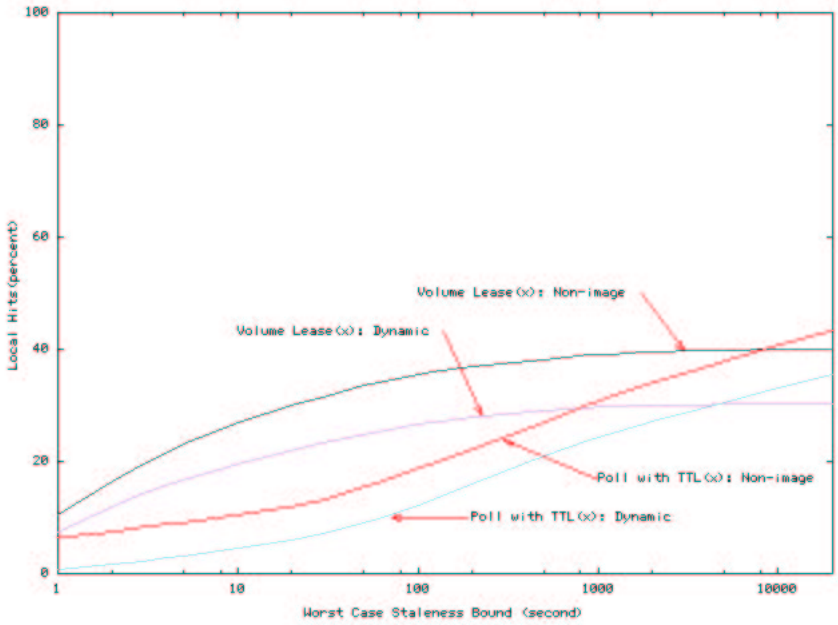

Abbildung 7: Vergleich erfolgreicher Cachezugriffe für dynamische Inhalte mit Volumen- und TTL-Fristen

Literatur

[IBM97] International Business Machines (Hrsg.). Olympic-Caliber Computing, o.O., 1997.

[Iyengar99] Iyengar, A. et al. Analysis and characterization of large-scale Web server access patterns and performance, o.O., 1999.

[Iyengar01] Iyengar, A. et al. Engineering Server-Driven Consistency for Large Scale Dynamic Web Services, Hong Kong, 2001.

[Mogul96] Mogul, J. A Design for Caching on HTTP 1.1, o.O., 1996.