Local Latency Reduction on the Web: Compressing, Caching and Prefetching

Seminar „Quality of Internet Services in E-Commerce“
Wintersemester 2000/2001

Axel Beckert, abe@cs.uni-sb.de

Betreuer: Prof. Gerhard Weikum

17. Februar 2000

Übersicht

1 Motivation

Hohe Wartezeiten und niedrige Byteraten im Web können verschiedene Gründe haben:

1. Modem-Verbindung oder andere schmalbandige Verbindung des Clients zum Rest des Internet
2. Temporär ausgefallene Router oder Leitungen und dadurch Überlastungen auf den Ausweichstrecken
3. Überlastete Server

Lokale Gegenmaßnahmen:

1. Datenvolumen der Modem-Verbindung verringern
2. Chaching mittels einem oder mehreren Proxies

2 Ansatzpunkte zur Traffic-Reduktion zwischen Client und Proxy

- Großer Cache auf Client-Seite
- Filterung von nicht gewünschten Inhalten durch den Benutzer
- $\triangle$-Komprimierung [Mogul, Douglis, Feldmann & Krishnamurthy, 1997]
- Komprimierung von HTML auf Anwendungsebene [Mogul et al., 1997]
- Durch Compilierung komprimierte Markup-Sprachen (z. B. WAP WMLC)
- Client-Initiated Prefetching (z. B. „Webbeschleuniger“)
- Proxy-Initiated Prefetching (Pushing)
- Proxy-Suggested Prefetching via HTTP Request Piggyback

2.1 Komprimierung

2.1.1 $\triangle$-Komprimierung

Braucht sehr großen Cache auf beiden Seiten:

- lokal, da eine ältere Version zum Vergleichen da sein muß und
- remote, da sowohl die aktuelle als auch (ggfs. mehrere) ältere Versionen — gegebenenfalls als „diff" gespeichert — vorhanden sein müssen.

Reduziert das Übertragungsvolumen geänderter Objekte um durchschnittlich 88% [Mogul et al., 1997] und schafft bis zu 14,6% Verzögerungsverringerung.

2.1.2 HTTP Transfer-Encoding

HTTP/1.1 unterstützt neben der Übertragung komprimiert gespeicherter Dokumente (via *Content Negotiation*) auch die on-the-fly Komprimierung nur für die Übertragung der Daten (HTTP-Header `Transfer-Encoding`). Übliche Komprimierungsverfahren sind hier „deflate" und „gzip".

Reduziert das Übertragungsvolumen um durchschnittlich 25% [Mogul et al., 1997].

2.1.3 Compilierte (binäre) Versionen von Markup-Sprachen

Da es sich bei HTML- und XML-Dokumenten um strukturierten Text mit fest definierten Tags und anderen häufigen Zeichenketten (z. B. `http://www.`) handelt, läßt er sich gut in eine strukturierte, binäre Form compilieren.

Aktuelle Anwendung: WAP-Seiten werden in compilierter Form (Compiled Wireless Markup Language, WMLC) übertragen und interpretiert, da WAP-Clients üblicherweise eine sehr schmalbandige Anbindung und nur einen sehr kleinen Cache besitzen.

2.2 Proxy-Initiated Prefetching [Fan, Cao, Lin & Jacobson, 1999]

Vorteile gegenüber Client-Initiated Prefetching:

- Die Vorhersage basiert auf statistischen Daten über alle Benutzer des Proxys.
- Das angefragte Dokument muß zur Vorhersage nicht geparst werden.
- Prefetching wird nur auf Dokumente angewendet, die der Proxy im Cache hat. Somit wird durch Prefetching ausschließlich im LAN zusätzlicher Traffic erzeugt, nicht aber in WANs.

Es gibt zwei Varianten:

- Prefetching mit Benachrichtigung über Hits im Cache des Browsers (bzw. des lokalen Proxys)
- Prefetching ohne Benachrichtigung über Hits im Cache des Browsers (bzw. des lokalen Proxys)

2.2.1 PPM Prefetching Algorithmus

Der von Fan et al. (1999) vorgeschlagene PPM-Algorithmus (Prediction by Partial Matching) kennt folgende Parameter:

- *PrefixDepth*: Anzahl der vergangenen Zugriffe, die zur Analyse verwendet werden
- *SearchDepth*: Anzahl der Zugriffe, die der Algorithmus versucht, in die Zukunft zu schauen
- *Threshold*: Wahrscheinlichkeit, ab der Prefetching veranlaßt wird

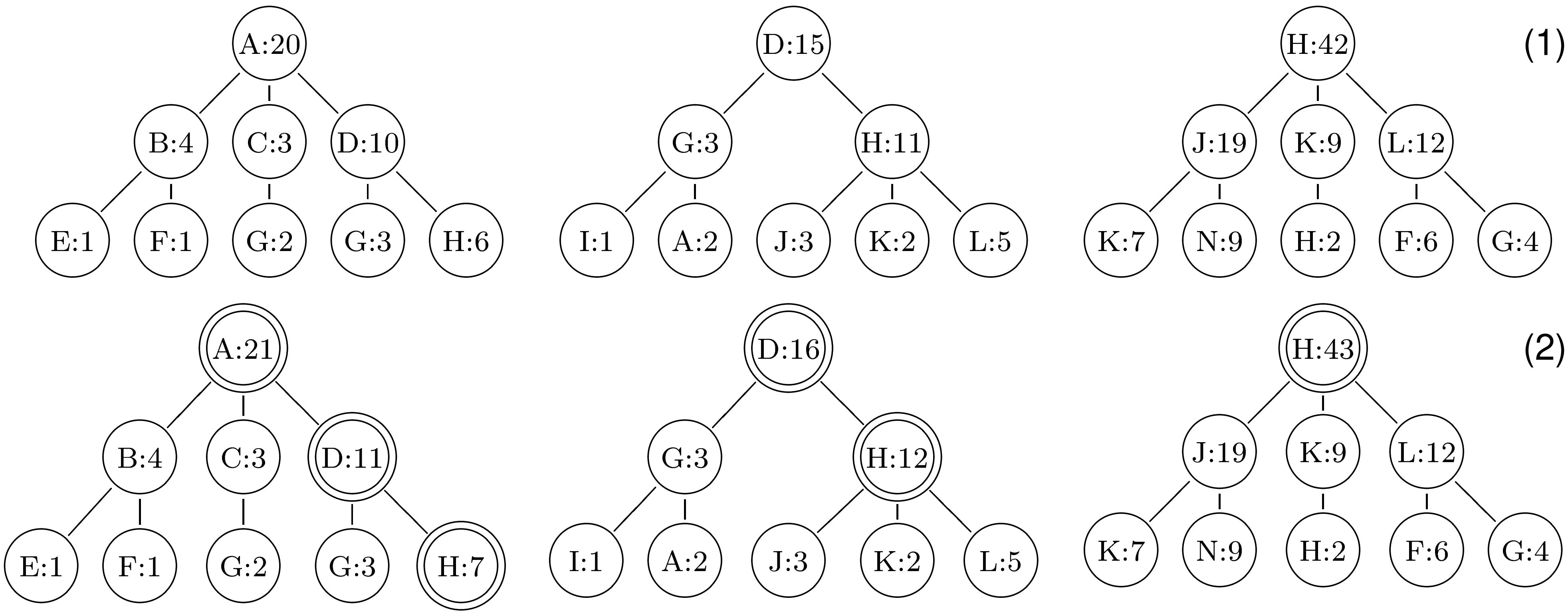
Funktionsweise:

1. Der Algorithmus vergleicht das letzte, die beiden vorletzten, ... und die *PrefixDepth* letzten zugegriffenen Dokumente des Benutzers mit den statistischen Daten eine entsprechende Datenstruktur und berechnet die Wahrscheinlichkeit aller Dokument, die nach den o. g. Sequenzen innerhalb von *SearchDepth* Schritten vorkamen.
2. Unter diesen Dokumenten werden diejenigen herausgesucht, deren Wahrscheinlichkeit über *Threshold* liegt und die kleiner als 50kB sind.
3. Die so gefundenen Dokumente werden absteigend nach der Länge des gematchten Präfixes sortiert, innerhalb einer Präfixlänge nach der Wahrscheinlichkeit.

Datenstruktur der für PPM notwendigen History

Für jedes Dokument im Cache existiert ein Baum der Tiefe *K=PrefixDepth+SearchDepth*, in dem alle *K* Zugriffe nach einem Zugriff auf das entsprechende Dokument gespeichert wird.

Beispiel: Bäume vor (1) und nach (2) dem Zugriff auf die Dokumentsequenz (*A*, *D*, *H*) bei *K*=3.



2.2.2 Potential verschiedener Konfigurationen des PPM-Algorithmus

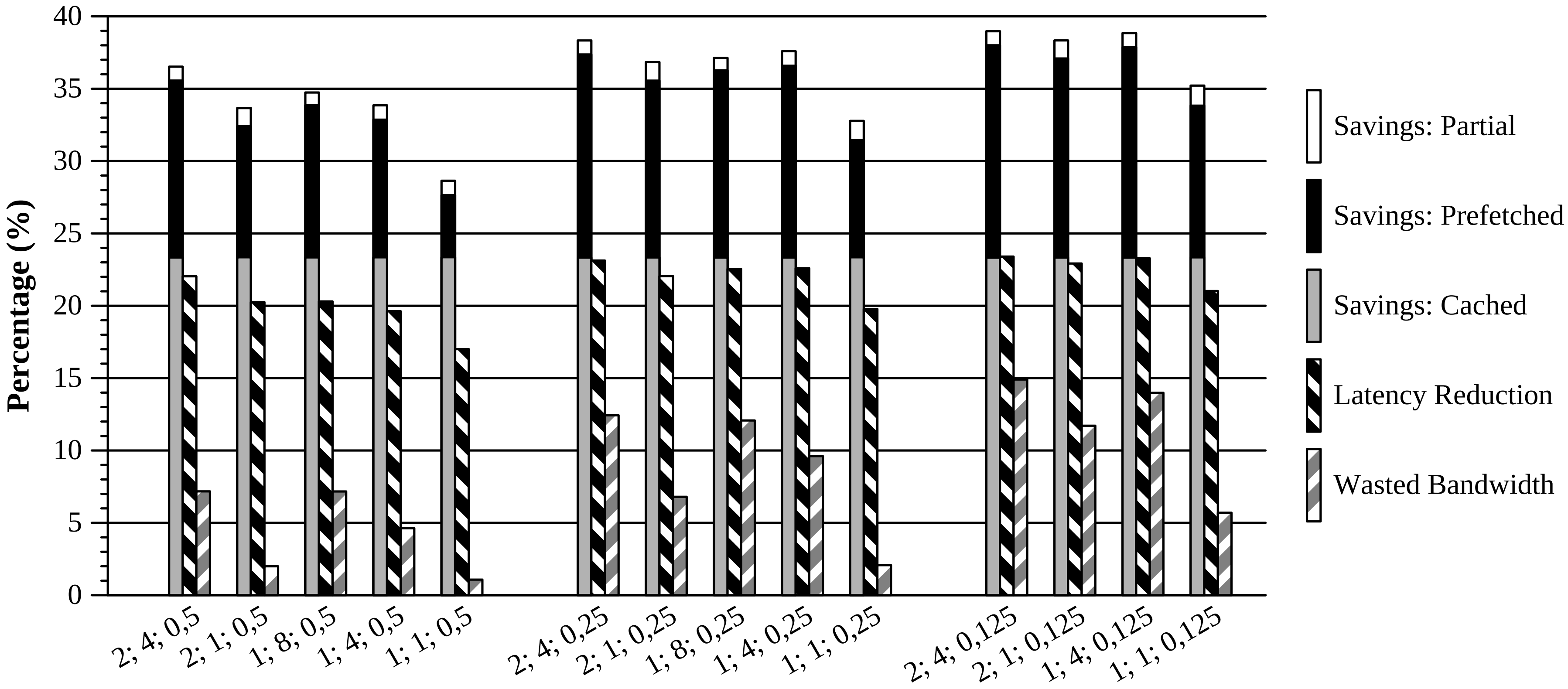


Abbildung 1: Anfrage-Einsparungen, Verzögerungsverringerung und verschwendete Bandbreite des PPM-Algorithmus. Konfigurationen des PPM-Algorithmus dargestellt als Tupel (*Prefix-Depth*, *SearchDepth*, *Threshold*).

2.2.3 Potential von Prefetching

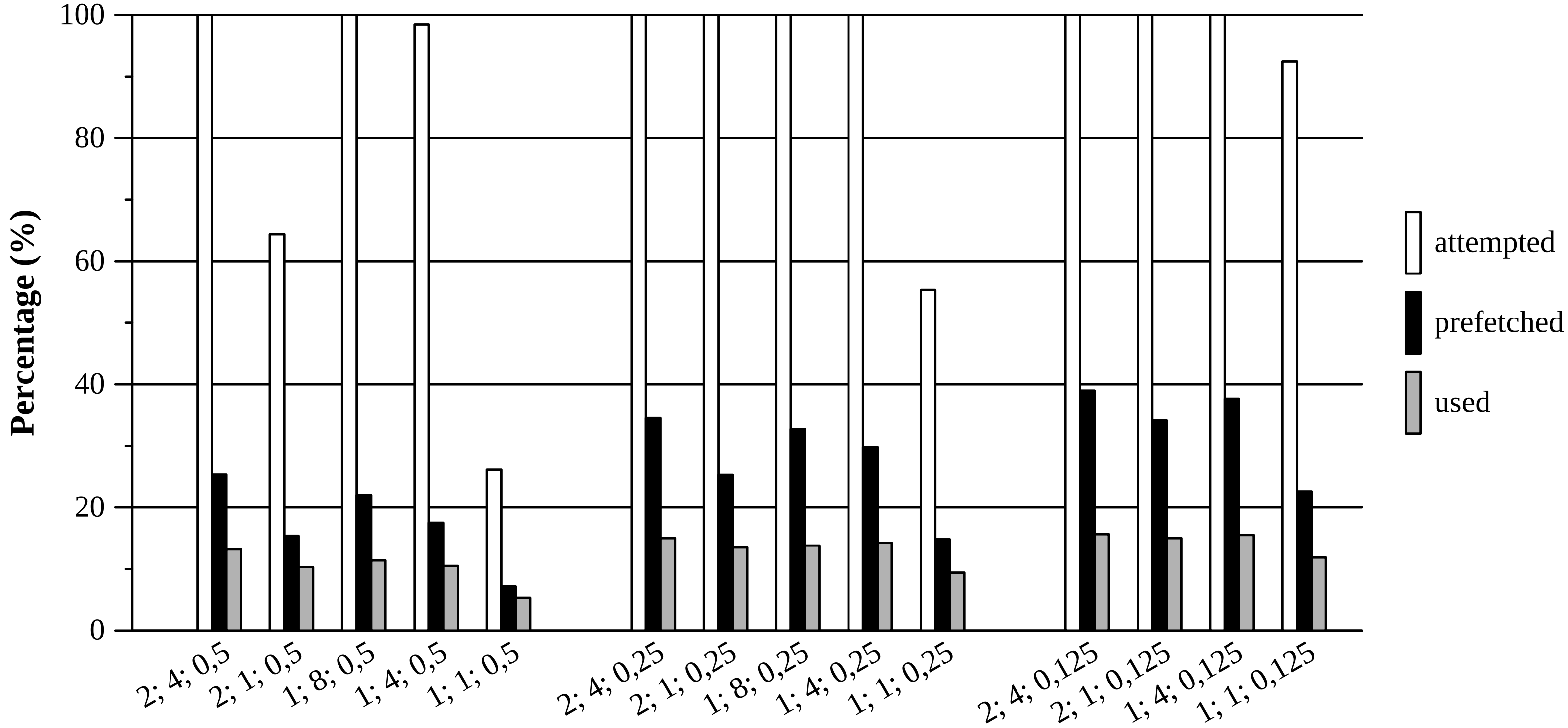


Abbildung 2: Gesamtanzahl an vorgeschlagenen Kandidaten (*attempted*), Anzahl der durchgeführten Prefetching-Vorgänge (*prefetched*), Anzahl der Zugriffe auf Objekte, die per Prefetching in den Browser-Cache kopiert wurden (*used*). Dabei entsprechen 100% der Gesamtanzahl von Anfragen im Trace. Konfigurationen des PPM-Algorithmus dargestellt als Tupel (*PrefixDepth*, *SearchDepth*, *Threshold*).

2.2.4 Aufschlüsselung von *prefetched* nach Dateitypen

- Ca. 48% GIFs
- Ca. 21% JPEGs
- Ca. 15,5% HTML-Dateien
- Rest andere oder anhand der Traces nicht identifizierbare Dateitypen

Dies legt nahe, daß es sich um eingebettet Grafiken der entsprechenden Webseiten handelte.

2.2.5 Aufschlüsselung der Genauigkeit nach Dateitypen

Ca. 65% GIFs

Ca. 58% JPEGs

Ca. 35% HTML- und andere Dateien

2.2.6 Auswirkungen von vereinfachten Implementationen

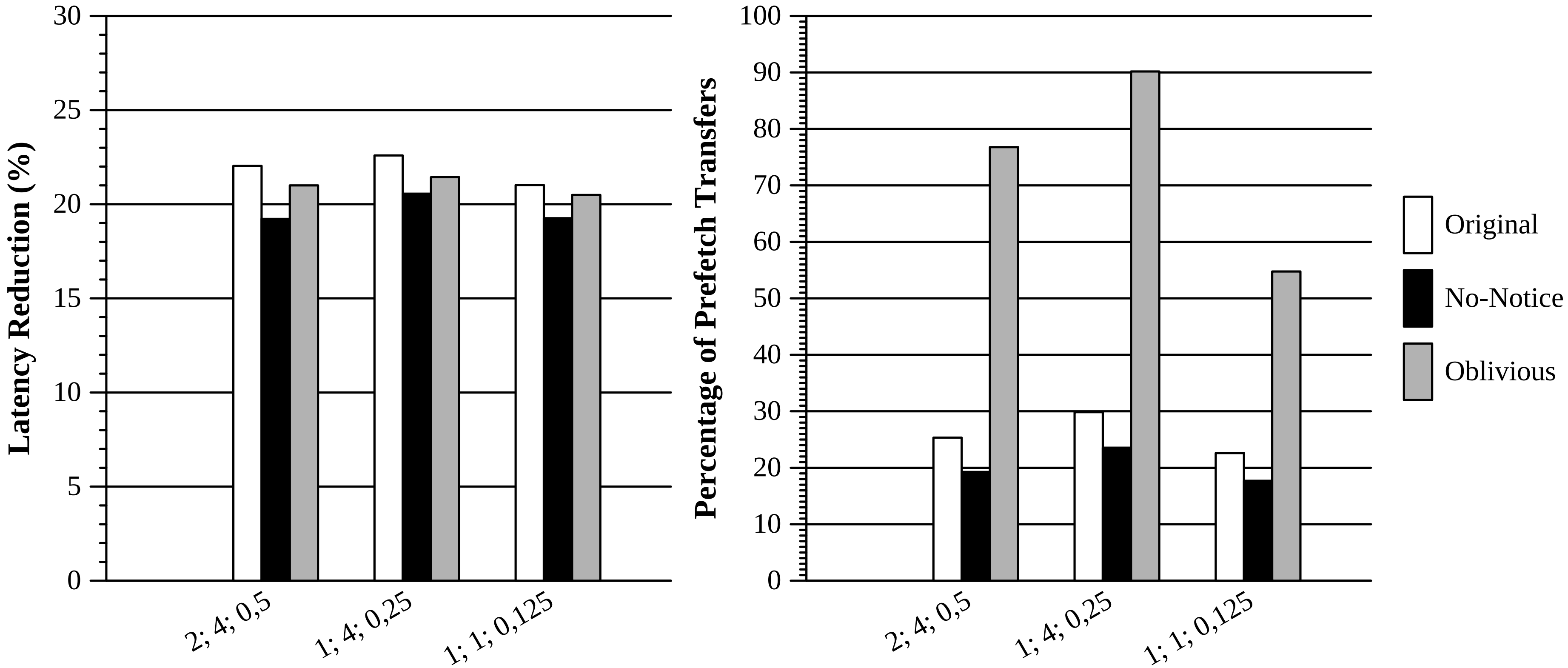


Abbildung 3: Vergleich des ursprünglichen Prefetchings mit Vereinfachungen. Konfigurationen des PPM-Algorithmus dargestellt als Tupel (*PrefixDepth*, *SearchDepth*, *Threshold*).

Auswirkungen von vereinfachten Implementationen (2)

Keine Benachrichtigung über Hits im Browser-Cache („No-Notice“)

Grund: Wesentlich einfacher zu implementieren.

Effekt: Prefetching wird weniger häufig initiiert.

Kein Wissen über Inhalt des Browser-Caches („Oblivious“)

Grund: Um das unnötiges Prefetching von Dokumenten zu verhindern, die bereits im Browser-Cache liegen, sollte der Proxy über den Cache-Inhalt aller seiner Clients Bescheid wissen. Dies ist aber aufgrund des zusätzlichen Speicherverbrauchs als auch von der Implementation her schwierig zu handhaben.

Effekt: Es werden wesentlich mehr Dokumente per Prefetching zum Client übertragen.

Alternative: Proxy-Suggested Prefetching via HTTP Request Piggyback

3 Web Proxy Caching

Caching mittels Web Proxy reduziert einerseits die Verzögerung von häufig angefragten Daten, andererseits das Datenvolumen der Anbindung des ISP zum Internet. Dazu tragen u. a. folgende HTTP-Header bei:

Pragma: Cache oder No-Cache

ETag: Hash-Wert eines Dokumentes, sollte innerhalb eines Servers eindeutig sein

Expires: Angabe über die Dauer der Gültigkeit eines Dokumentes

Last-Modified: Angabe über Datum der letzten Änderung.

Dynamische Dokumente oder Dokumente, die via *Content Negotiation* ausgesucht werden, machen Probleme beim Caching. Hierbei helfen außerdem noch folgende HTTP-Header:

Vary: z. B. Encoding, Type, Language

Content- & Accept-Type: z. B. text/plain, text/enriched, text/html, text/vnd.wap.wml, application/pdf, etc.

Content- & Accept-Encoding: z. B. gzip

Content- & Accept-Language: z. B. de, en, fr, dk, nl, no

3.1 Cooperative Web Proxy Caching

Cooperative Web Proxy Caching ist die Zusammenarbeit von zwei oder mehreren Web Proxy Caches. Grundgedanke: Es kann immer noch schneller sein, seinen Cache-Kollegen zu fragen, als den eigentlich angefragten Server.

Wolman et al. (1999) versuchen anhand Auswertung von gleichzeitigen Traces an der *University of Washington* und *Microsoft* die Nützlichkeit von *Cooperative Web Proxy Caching* in der Praxis zu ermitteln. Sie kamen dabei zu folgenden Ergebnissen:

- Cooperative Web Caching ist nur für kleine Organisationen (bis ca. 2500 Nutzer) nützlich und liegt damit in einem Bereich in dem es technisch (aber eben meist nicht politisch oder geographisch) auch ein einzelner, großer Proxy Cache täte.

- Bei großen Organisationen reduziert Cooperative Web Caching die Verzögerung nur marginal: „Unbeliebte Dokumente sind universell unbeliebt."

Cooperative Web Proxy Caching (2)

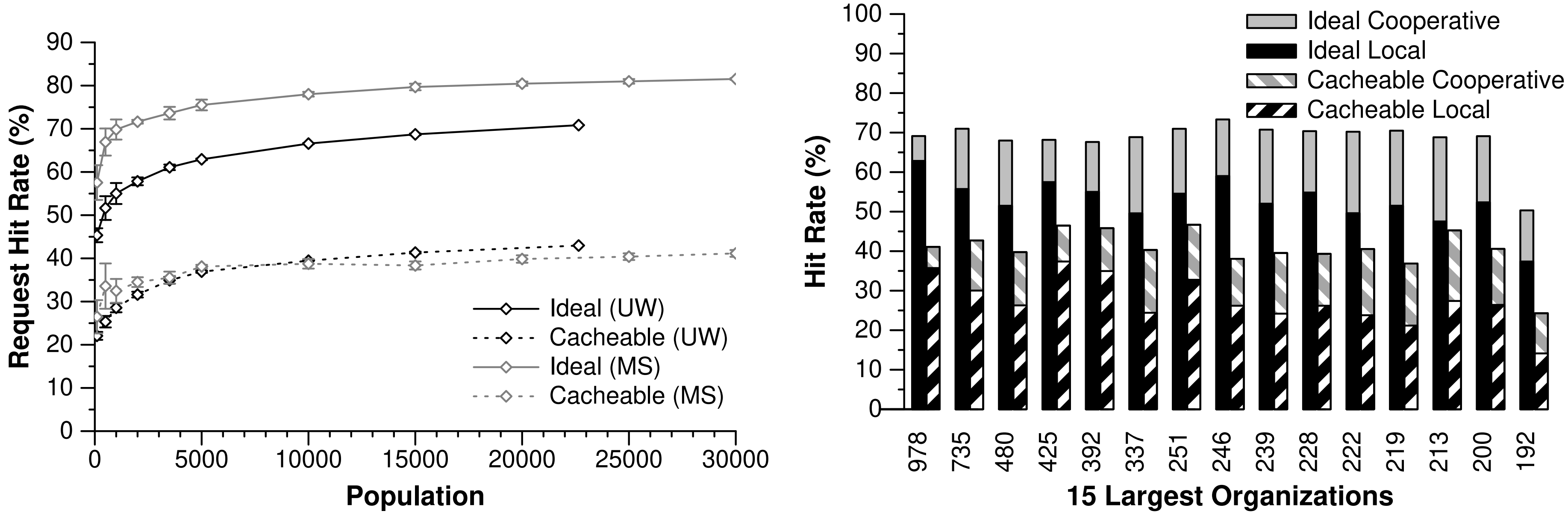


Abbildung 4: Proxy Anfrage-Trefferquoten als Funktion der Größe der Benutzerpopulation sowie lokale und globale Proxy Anfrage-Trefferquoten für die 15 größten Suborganisationen der *University of Washington*

3.2 HTML Macro-Preprocessing to Support Dynamic Document Caching

[Douglis, Haro & Rabinovich, 1997]

- Dynamische Dokumente sind abhängig von mindestens einem der folgenden Faktoren:
  - Datum
  - Uhrzeit (z. B. Wettervorhersage, Nachrichten)
  - Cookies (z. B. viele Online-Shops, Slashdot & Co.)
  - Referer-Header
  - lokalen Daten(banken) auf dem Server

  Dementsprechend können sie normalerweise nicht im Cache gespeichert werden.

- In vielen Fällen hilft das Offenlegen der internen Struktur dynamischer Seiten, die statischen Elemente dieser Seiten (Layout von Tabellen, Menüs, Kopf- und Fußzeilen, etc.) zu cachen, so daß nur noch die dynamischen Daten vom Server geholt werden müssen.

4 Literatur

Douglis, F., Haro, A. & Rabinovich, M. [1997]. „HTML Macro-Preprocessing to Support Dynamic Document Caching.“ In *Proceedings of USENIX‘97.* (Available from http://www.research.att.com/˜douglis/)

Fan, L., Cao, P., Lin, W. & Jacobson, Q. [1999]. „Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performances.“ In .

Fielding, R., Gettys, J., Mogul, J. C., Frystyk, H. & Berners-Lee, T. [1997, Jan]. *RFC 2068: Hypertext Transfer Protocol – HTTP/1.1.*

Mogul, J. C., Douglis, F., Feldmann, A. & Krishnamurthy, B. [1997]. „Potential benefits of delta encoding and data compression for http.“ In *Proceedings of ACM SIGCOMM‘97.* (Available from http://www.research.att.com/˜douglis/)

Wolman, A., Voelker, G. M., Sharma, N., Cardwell, N., Karlin, A. & Levy, H. M. [1999]. „On The Scale and Performance of Cooperative Web Proxy Caching.“ In *Operating Systems Review* (Vol. 34, Seite 16-31).