# Speculative dissemination and service for web content delivery (a.k.a., predicting the birth...

# Speculative Data Dissemination and Service
to Reduce Server Load, Network Traffic and Service Time
in Distributed Information Systems

*Azer Bestavros*

Computer Science Department
BOSTON UNIVERSITY

Monday, March $11^{th}$, 1996

*IEEE ICDE'96, New Orleans, Louisiana*
*February $28^{th}$, 1996*

# Speculative Data Dissemination and Service
to Reduce Server Load, Network Traffic and Service Time
in Distributed Information Systems

*Azer Bestavros*

Computer Science Department
BOSTON UNIVERSITY

Wednesday, February $28^{th}$, 1996

† This work is supported partially by a grant from the NSF.

# Locality of Reference in a Client-Server Environment

## Locality of Reference Flavors

◇ **Temporal**:
A document accessed frequently in the past is likely to be accessed again in the future.

◇ **Spatial**:
A document "neighboring" a recently accessed document is likely to be accessed in the future.

◇ **Geographical**:
A document accessed by a client is likely to be accessed in the future by "neighboring" clients.

## How to Capitalize on it?

◇ On the client side, use "caching" and "prefetching" (e.g. Distributed file systems, Sun NFS, AFS, [Standberg 1985, Morriss 1986, Howard 1988], Proxy caching [Danzig 1993, Acharya 1993, Papadimitriou 1994], Cooperative client caching [Blaze 1993, Dahlin 1994]).

◇ On the server side, use "information dissemination" [Bestavros 1994], "geographical caching" [Braunh and Claffyh 1994], "speculative service" [Bestavros 1995], "geographical push caching" [Gwertzman and Seltzer 1995].

# Information Caching versus Information Dissemination

## Motive

- ◇ The scalability of Internet services hinges on efficient distribution and partitioning of system resources to reduce the amount of data that must be moved.

## Information Caching

- ◇ Initiated by a client or a group of clients.
- ◇ Geared towards reducing service time.
- ◇ Relies on temporal locality of client reference patterns.
- ◇ Ensuring consistency is expensive.

## Information Dissemination

- ◇ Initiated by servers.
- ◇ Geared towards balancing load and reducing traffic.
- ◇ Relies on temporal/geographical popularity of documents.
- ◇ Ensuring consistency is cheap.
- ◇ Requires collaboration of "server proxies".

# Client-initiated Caching Study

## Experiment Description

- ◇ We instrumented Mosaic to log all user accesses on our site [BCC:95].
- ◇ We studied cache performance at various levels:
  - Session Caching: One cache per session
  - Host Caching: One cache per host
  - LAN Caching: One cache per LAN
- ◇ We used the logs obtained from Mosaic to perform trace simulations for various protocols [BCCCHM:95].

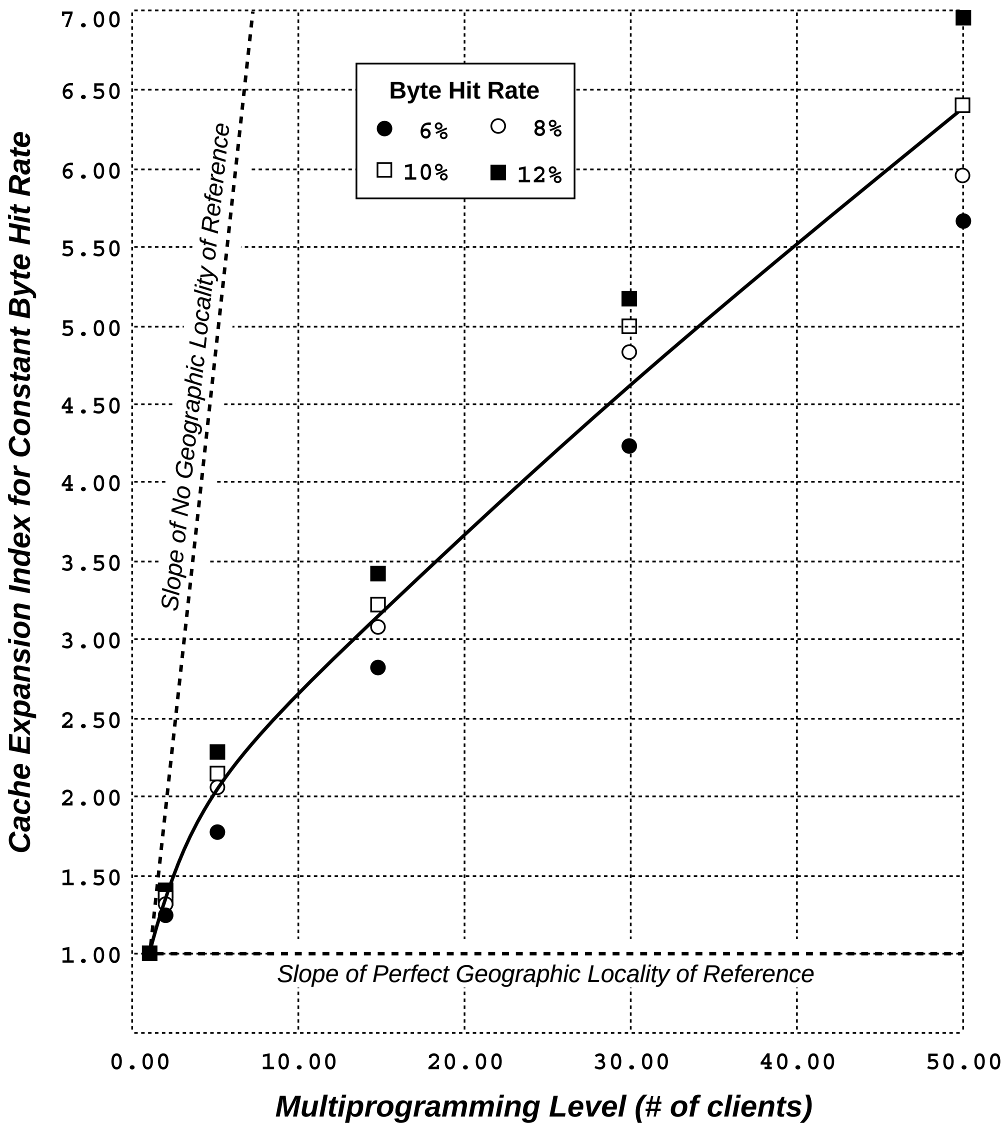
| | |
|---|---|
| Sessions | 4,700 |
| Users | 591 |
| URLs Requested | 575,775 |
| Files Transferred | 130,140 |
| Unique Files Requested | 46,830 |
| Bytes Requested | 2713 MB |
| Bytes Transferred | 1849 MB |
| Unique Bytes Requested | 1088 MB |

Summary Statistics for Trace Data Used in This Study

# Client-initiated Caching Effectiveness

## Experiments Results

- ◇ Poor Byte Hit Rate < 40% with infinite cache.
- ◇ Sharing amongst multiple clients is limited too!




Cache Expansion Index for Remote documents

# The Server's Perspective

## Server Log Analysis

- ◇ We collected the logs of our departmental HTTP server and those of the Rolling Stones Multimedia server.
- ◇ We used the logs to analyze the popularity of various documents and to drive trace simulations of various server-initiated protocols.

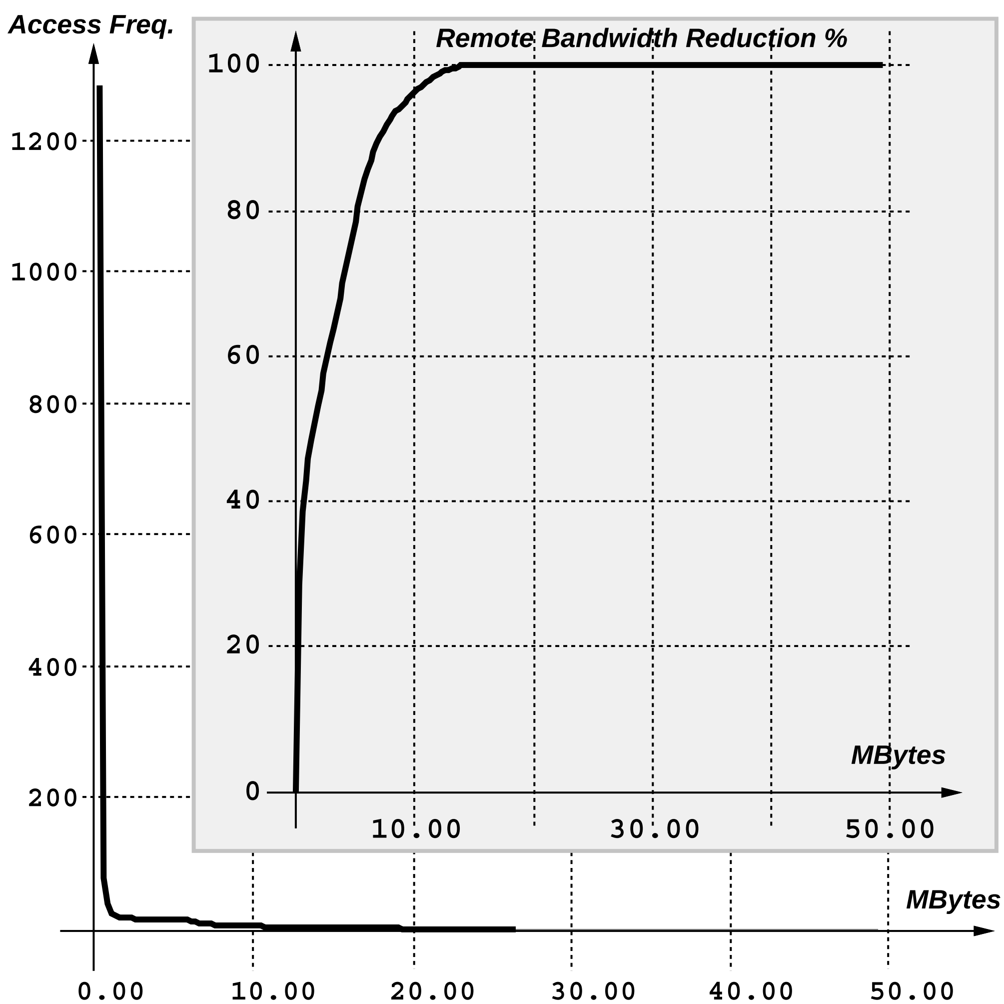
| | `cs-www.bu.edu` | `www.stones.edu` |
|---|---|---|
| Period | 56 days | 110 days |
| URL requests | 172,635 | 4,068,432 |
| Bytes transferred | 1,447 MB | 112,015 MB |
| Average daily transfer | 26 MB | 1,018 MB |
| Files on system | 2,018 | N/A |
| Files accessed (remotely) | 974 (656) | N/A (1,151) |
| Size of (accessed) file system | 50 MB (37 MB) | N/A (402 MB) |
| Unique clients (10+ requests) | 8,123 | 60,461 |

Summary Statistics for Log Data Used in This Study

## The Server's Perspective

**Log Analysis of** `http://cs-www.bu.edu`

- ◇ Popular documents are *very* popular!
- ◇ Only 10% of all blocks accounted for 91% of all requests!

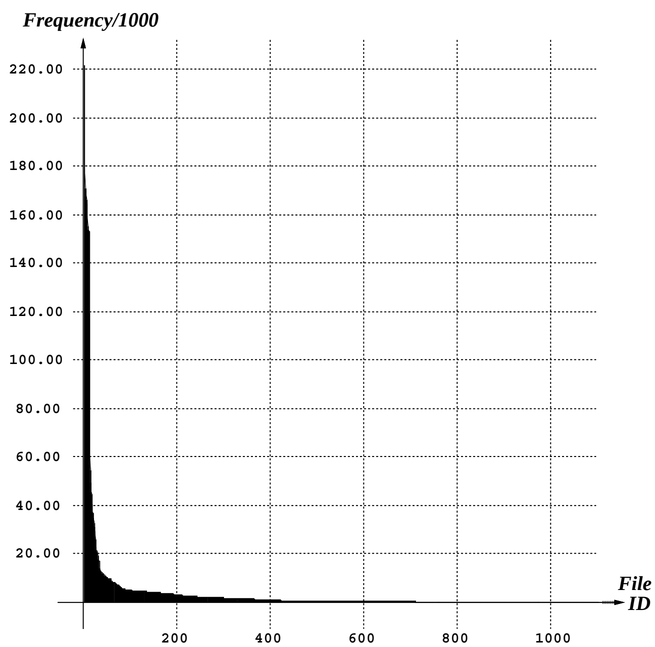



Popularity of data blocks and bandwidth reduction from their dissemination
`http://cs-www.bu.edu`

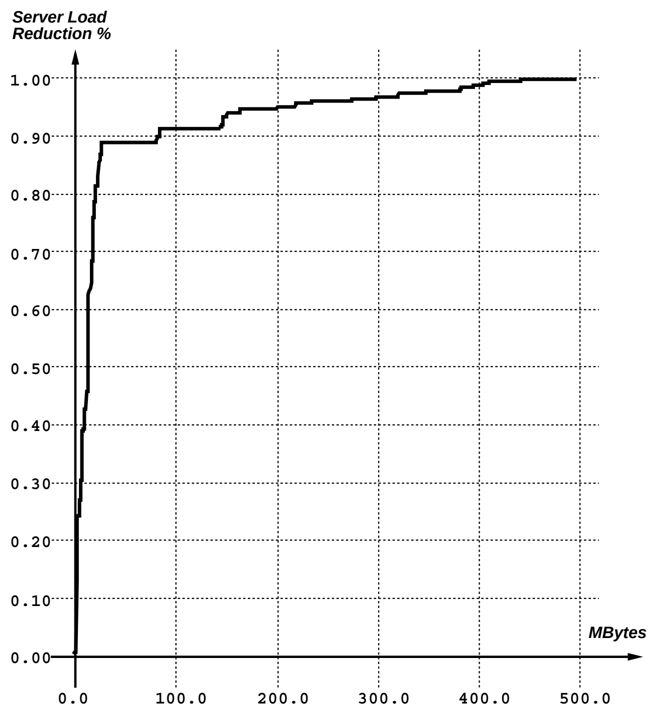
## The Server's Perspective

### Log Analysis of `http://www.stones.com`

- ◇ Same conclusions as before.
- ◇ Making 25MB of data available to clients at a proxy one-hop closer to them would save more than 900MB/day of network bandwidth.




Access frequency



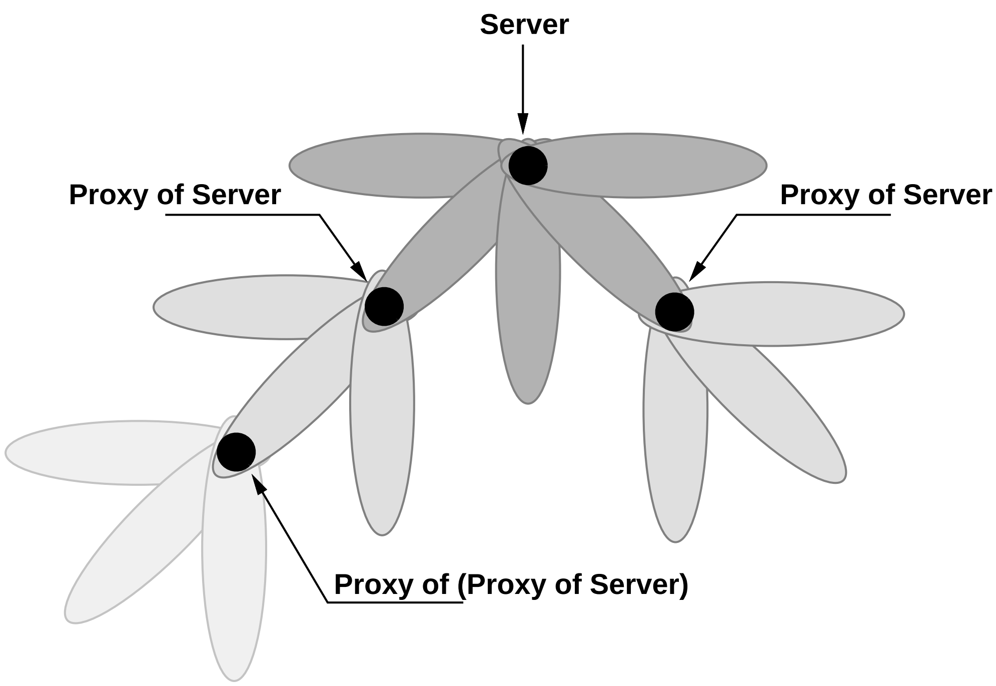

Expected bandwidth reduction

`http://www.stones.com`

# Information Dissemination Protocol

## Underlying Model

- ◇ A set of *service proxies* act as information "outlets" on the Internet.
- ◇ These *service proxies* offer space/bandwidth "for-rent" to other servers or proxies that constitute its *Cluster*.
- ◇ A server may belong to several clusters, thus allowing some of its files to be dessiminated to multiple service proxies.
- ◇ Service proxies are themselves servers who may be members of other clusters.




Underlying Model for Information Dissemination

# Information Dissemination Protocol

## Questions to be answered

- ◇ Given the access pattern at a server, *which clusters should the server choose to join?*
- ◇ Given the access pattern at a server, *which files should the server disseminate? and where?*
- ◇ Given the popularity profile of all servers in a cluster, *how should the resources (space/bandwidth) at the service proxy be allocated?*

## Assumptions

- ◇ The dissemination protocol should not require any "special" features/capabilities from other protocols.
- ◇ File popularity is a "universal" phenomenon (i.e. the probabilty of accessing a file is independent of who is accessing it). This is a *conservative* simplifying assumption.
- ◇ File popularity does not change drastically in a short period of time. This assumption has been verified.

# Optimal Allocation of Storage at the Proxy

## Notation

- ◇ $\mathcal{C} = \mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ is the set of servers in a cluster. $\mathcal{S}_0$ is the service proxy of $\mathcal{C}$.
- ◇ $R_i$ is the total number of bytes per unit time serviced by server $\mathcal{S}_i$ to clients outside $\mathcal{C}$.
- ◇ $H_i(b)$ is the probability that a request to $\mathcal{S}_i$ will be to the most popular $b$ bytes disseminated to $\mathcal{S}_0$.
- ◇ $B_i$ is the number of bytes that $\mathcal{S}_0$ duplicates from $\mathcal{S}_i$. $B_0 = B_1 + B_2 + \ldots + B_n$ is the total storage space available at $\mathcal{S}_0$.

## Goal

- ◇ Choose $B_i$ to maximize the percentage of traffic serviced at $\mathcal{S}_0$.

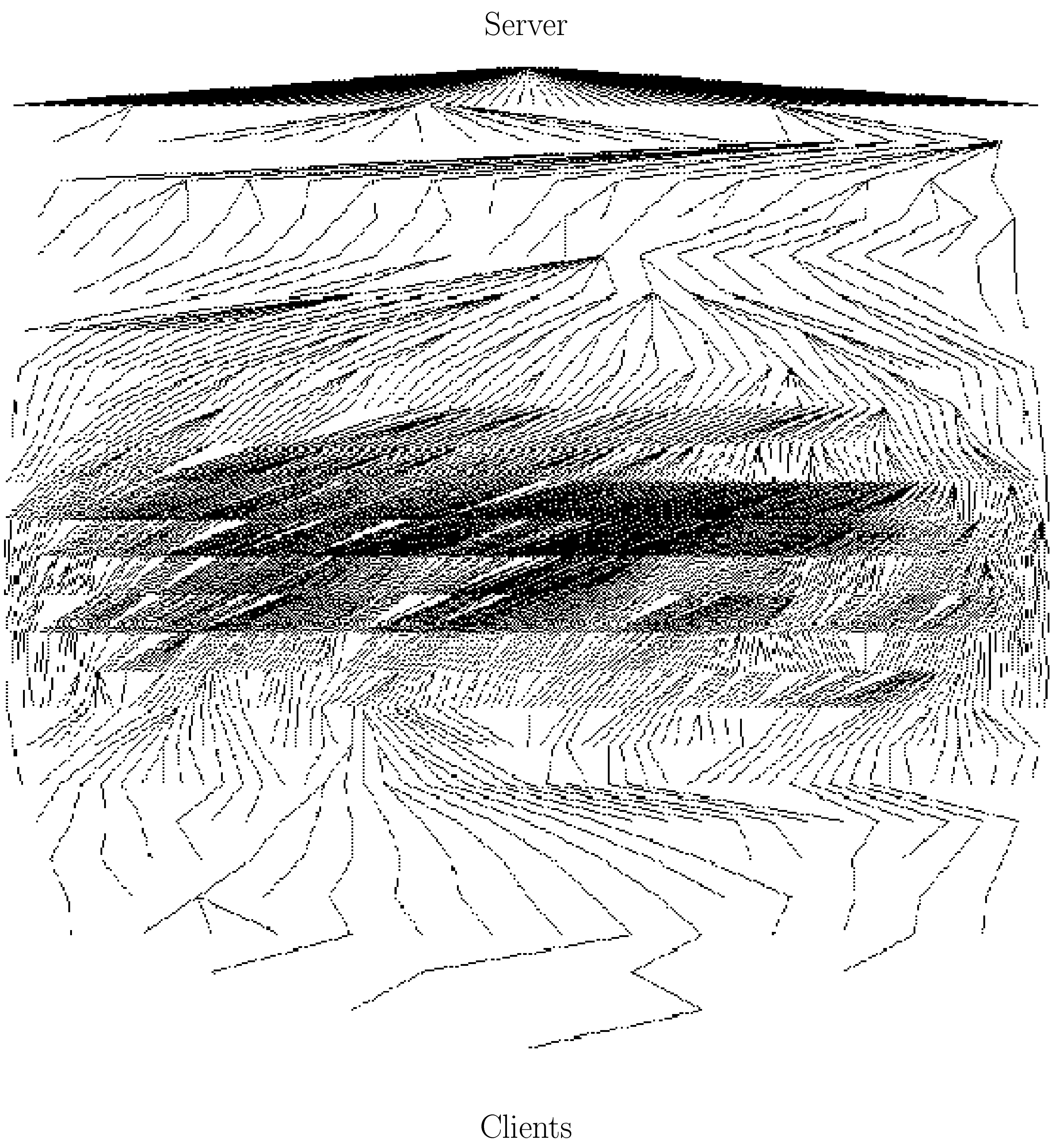
$$\alpha_{\mathcal{C}} = \frac{\Sigma_{i=1}^{n} R_i \times H_i(B_i)}{\Sigma_{i=1}^{n} R_i}$$

# Which Proxies Should be Contracted?

## Characterizing the Client Tree and Choosing Proxies

- ◇ Using the `record route` option of TCP/IP, it is possible to build a complete tree originating at the server with clients at the leaves. For `http://cs-www.bu.edu`, this tree consisted of 18,000 nodes.
- ◇ The most popular files are disseminated down the tree and stored at proxies *closer* to the clients.
- ◇ The location of such proxies depends on the demand from the various parts of the tree.
- ◇ Analysis of `http://cs-www.bu.edu` logs for a consecutive 26-week period suggests that the shape of the tree (especially internal nodes) and the distribution of load is quite static over time.

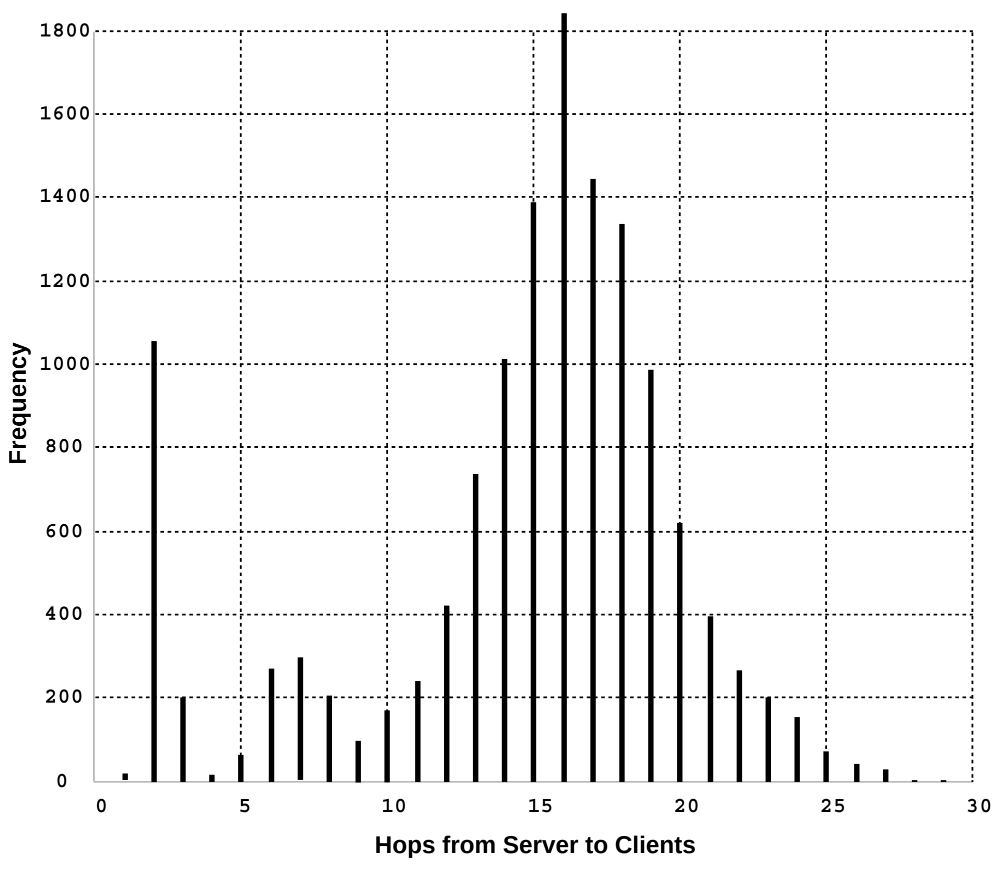
## How Does the Internet Look to a Server?



## How Much Bandwidth is Saved?

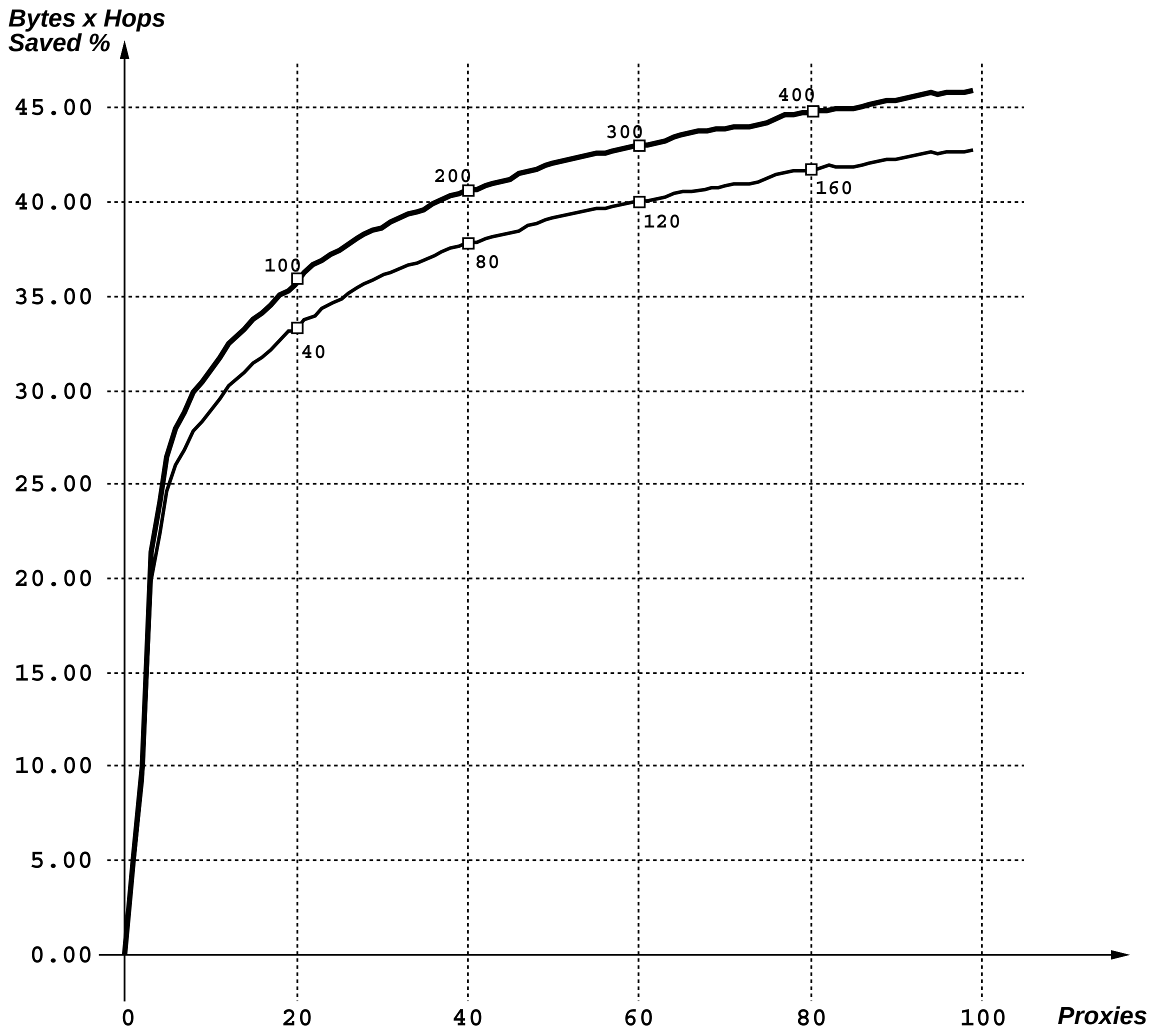
### How far could we “push” information towards clients?

- ◇ At least 8-9 hops!
- ◇ Replicating the most popular 25 MB from `http://www.stones.com` on *few* proxies yields a whoping saving of $>$ 8 GB of network bandwidth per day.




How far away are clients?

## How Much Bandwidth is Saved?




Expected reduction in bandwidth as a result of dissemination

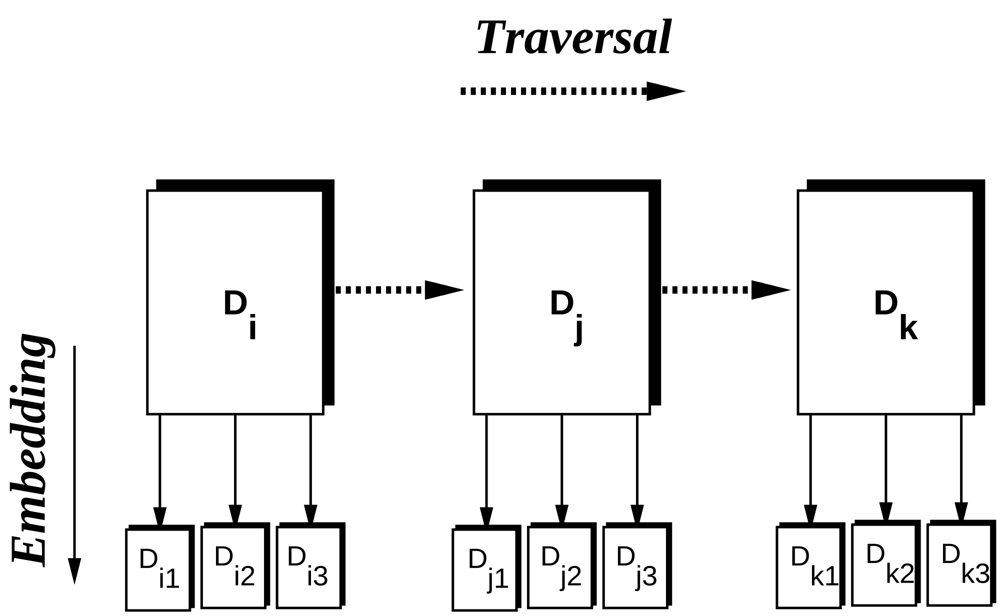
## The Notion of Speculative Service

### Could the next client request be predicted?

- ◇ In many cases, the answer is *yes*.
- ◇ Servers could "speculatively" service documents *before* they are requested (a.k.a. server-initiated prefetching).

### Two kinds of dependencies:

- ◇ Embedding dependencies: Document $\mathcal{D}_j$ is embedded in $\mathcal{D}_i$.
- ◇ Traversal dependencies: Document $\mathcal{D}_j$ is often requested as a result of an access to $\mathcal{D}_i$.




How far away are clients?

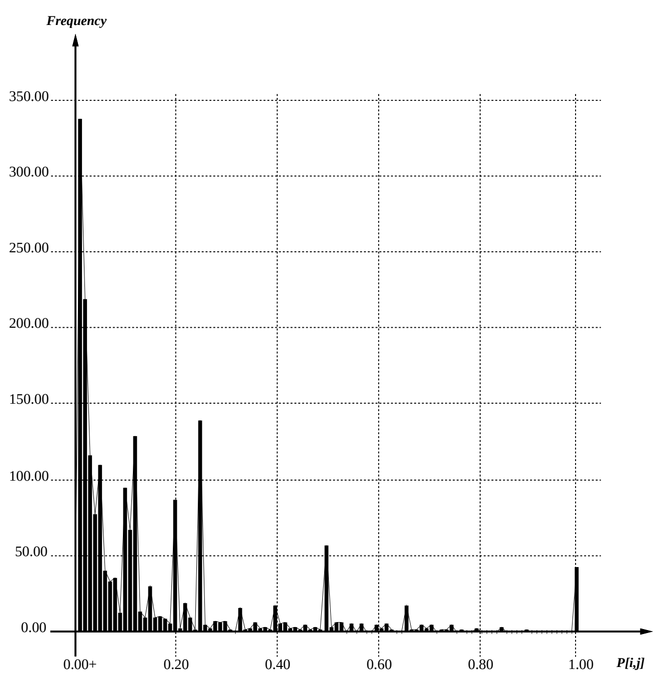
# Document Access Interdependency Matrix

## Notation

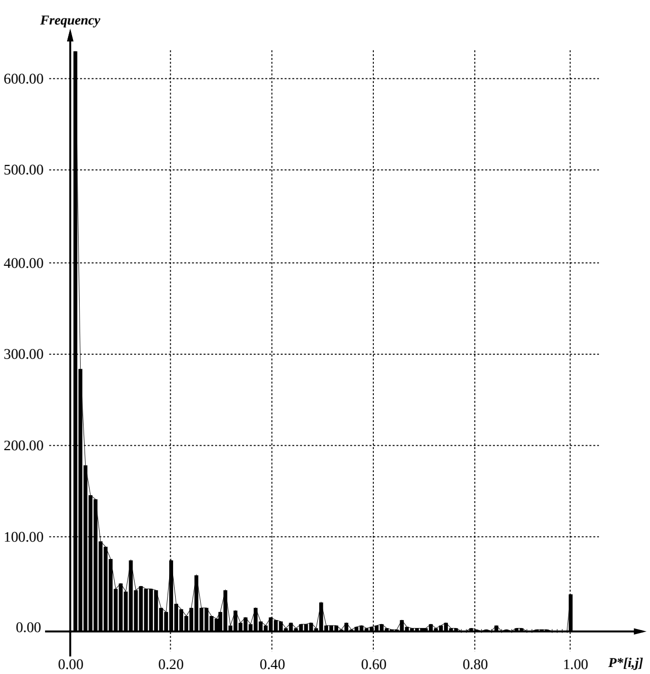
- ◇ Let $p[i, j]$ denote the conditional probability that document $\mathcal{D}_j$ will be requested, within $T_w$ units of time after the request for $\mathcal{D}_i$.
- ◇ Let $P$ denote the square matrix representing $p[i, j]$, for all possible documents $0 \geq i, j \leq N$. Let $P^*$ denote the transitive closure of $P$.
- ◇ Thus, $p^*[i, j]$ is the probability that there will be a sequence of requests (inter-request time ¡ $T_w$) starting with $\mathcal{D}_i$ and ending with $\mathcal{D}_j$.

## Server log analysis

- ◇ Using server logs, the $P$ and $P^*$ matrices could be easily constructed.




Histogram for ranges of $p[i, j]$




Histogram for ranges of $p^*[i, j]$

## Speculative Service Experiments

### Simulation Model

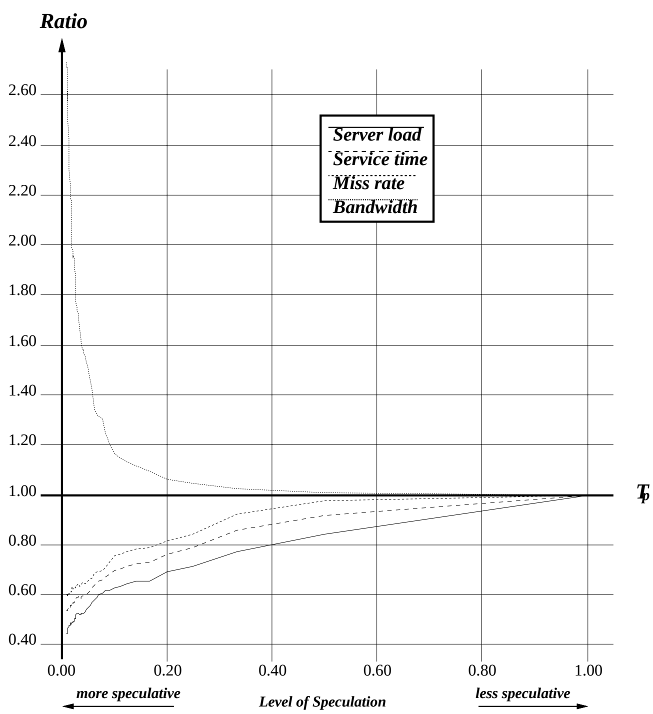
- ◇ Successive requests separated by less than `StrideTimeout` units of time belong to the same "stride".
- ◇ Clients maintain a session cache. The session cache is purged if the time between successive requests exceeds `SessionTimeout`.
- ◇ Four metrics are used: **`Bandwidth ratio`**, **`Server Load ratio`**, **`Service Time ratio`**, and **`Miss rate ratio`**.

| Parameter | Meaning | Base Value |
|---|---|---|
| `CommCost` | Cost of communicating 1 Byte | 1 unit |
| `ServCost` | Setup cost for a service request | 10,000 unit |
| `StrideTimeout` | Value of time window $T_w$ | 5.0 secs |
| `SessionTimeout` | Cache invalidation timeout | $\infty$ secs |
| `MaxSize` | Maximum size to prefetch | $\infty$ (no limit) |
| `Policy` | Speculative service algorithm | $p^*[i,j] \geq T_p$ |
| `HistoryLength` | Length of the logs used for $P$ | 60 days |
| `UpdateCycle` | Frequency of recomputing $P$ | 1 day |

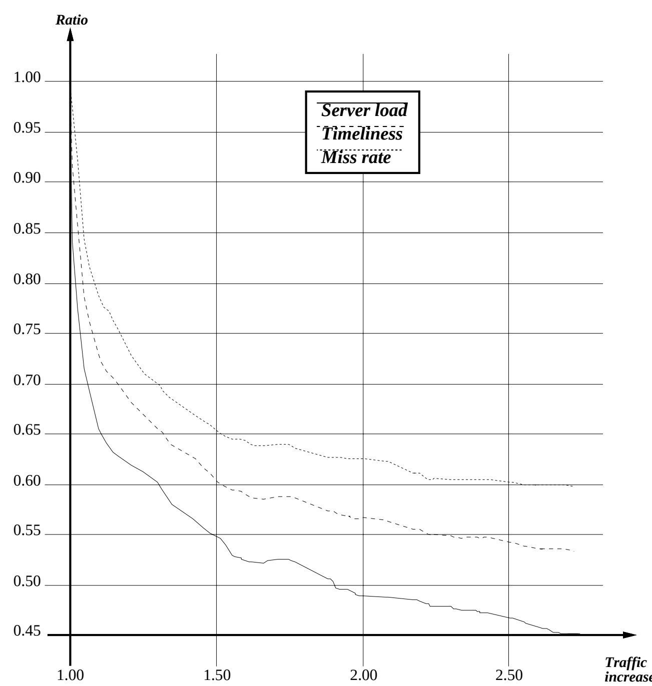
## Speculative Service Experiments

### Baseline Results

- ◇ Significant improvement in performance (above what is achievable by client caching) could be achieved for a miniscule increase in traffic.
- ◇ 5% extra bandwidth results in a whopping 30% reduction in server load, a 23% reduction in service time, and a 18% reduction in client miss-rate.
- ◇ Beyond some point, speculation does not seem to pay off.




Metrics vs Speculation Level

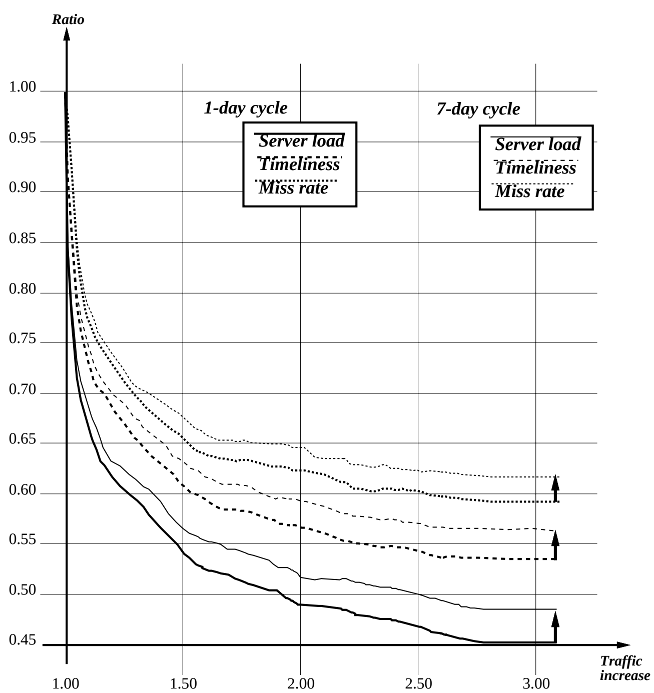



Metrics vs Bandwidth

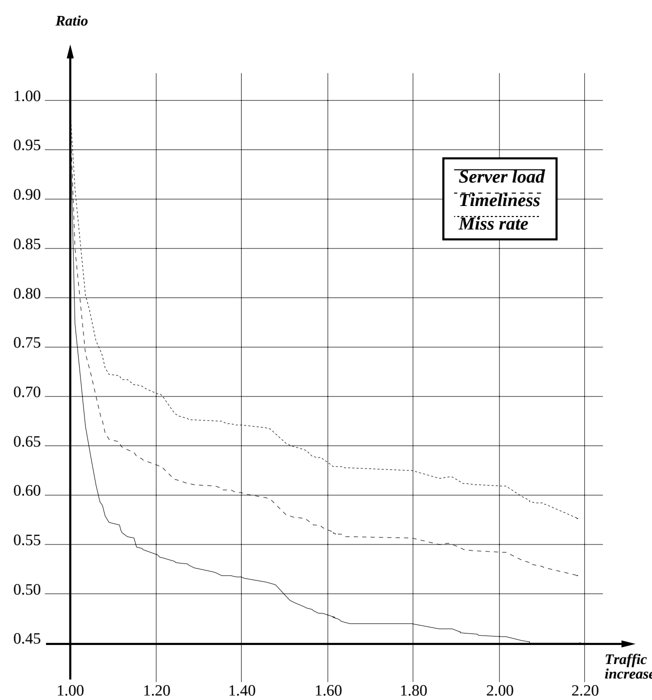
# Speculative Service Experiments

## Stability of the $P$ and $P^*$ Relations

- ◇ We varied the `UpdateCycle` from 1 to 7 days, while keeping the `HistoryLength` at 60 days. This change resulted in a 3% degradation in all measured metrics, suggesting that $P$ and $P^*$ do change (albeit very slowly) with time.
- ◇ Also, we varied the `HistoryLength` from 60 to 30 days, while keeping the `UpdateCycle` at 1 day. This change resulted in a 5% improvement in all measured metrics, suggesting that an aging mechanism must be used to phase-out dependencies exhibited in on older server traces, in favor of dependencies exhibited in more recent ones.




Effect of slower `UpdateCycle`

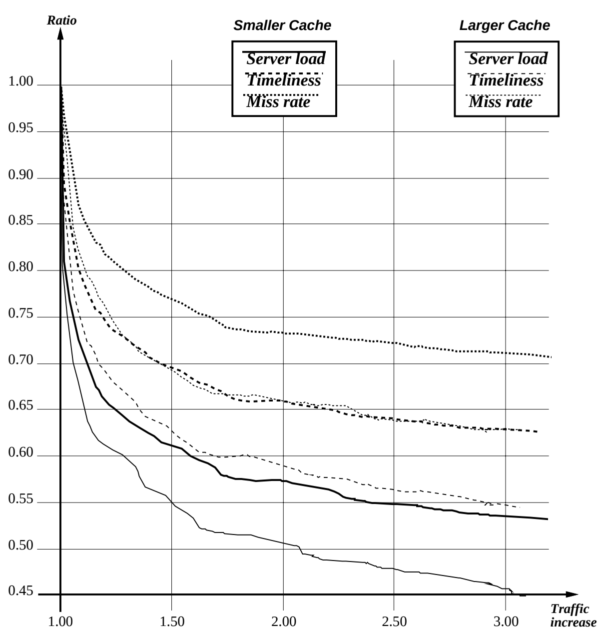



Effect of shorter `HistoryLength`

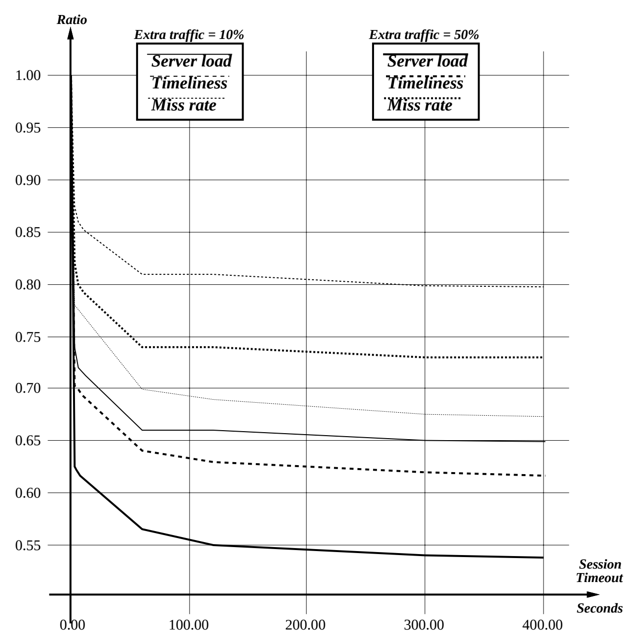
## Speculative Service Experiments

### Effect of Client Caching

- ◇ We compared simulations with `SessionTimeout` equal to 3,600 seconds (large cache) and to 120 seconds (small cache).
- ◇ The presence of client caching (even if modest) is likely to further improve the performance of speculative service.
- ◇ In order to reap all the benefit from speculative service, client must cache "prefetched" documents long enough.




Effect of client caching



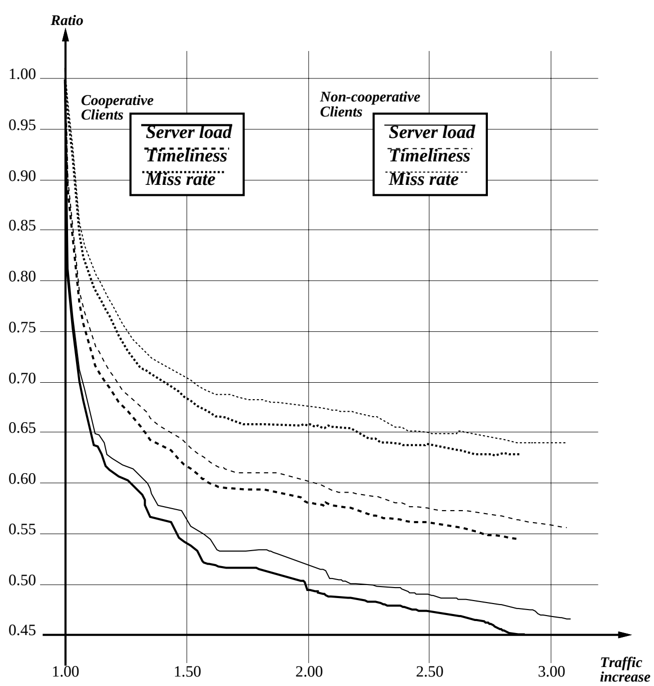

Effeciency of client caching

## Speculative Service Experiments

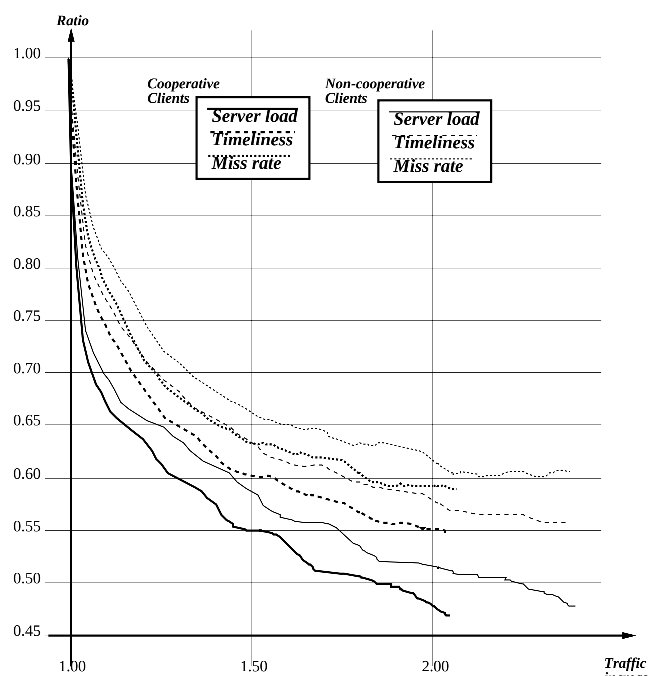
### Cooperative Clients

◇ Performance could be further improved if documents "already cached at the client" are not speculatively serviced!

◇ Our simulations showed that speculative service with cooperative clients results in better bandwidth utilization, especially when the client performs "some" caching.




Cooperative client (small cache)

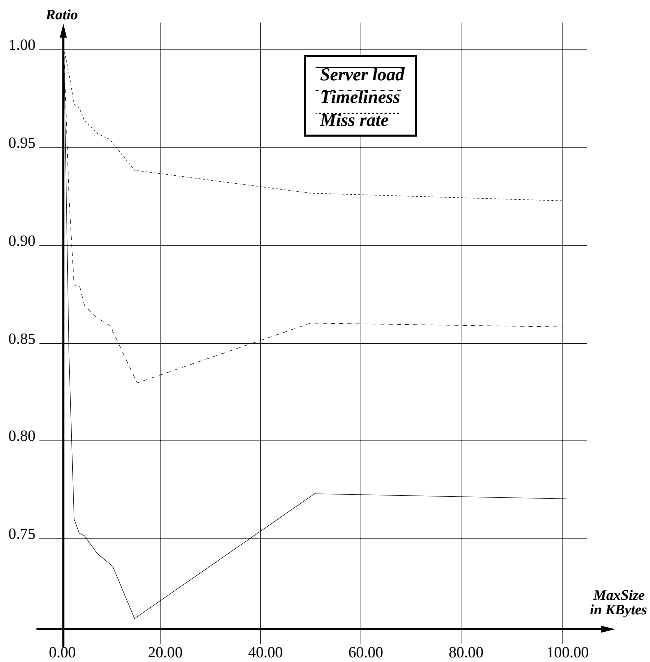



Cooperative client (infinite cache)

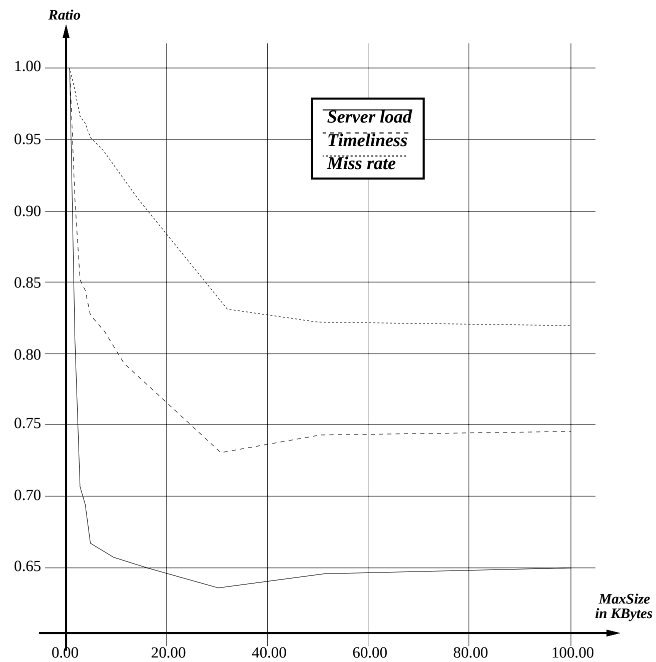
## Speculative Service Experiments

### Effect of Document Size

◇ The benefits of speculation are most pronounced when documents serviced speculatively are small. We studied this by varying the `MaxSize` parameter.




Low level of speculation




High level of speculation

# Variations on Speculative Service

## Server-assisted Prefetching

- ◇ Servers could pass the list of "probable future documents" to the client (instead of passing along the document themselves).
- ◇ Prefetching could be done at the discretion of clients.

## Client-initiated Prefetching

- ◇ Using user traces, it is possible for the client software to perform "prefetching" [Bestavros and Cunha: 1995].
- ◇ Client-initiated prefetching is *very* effective for "frequently traversed documents" but ineffective for "newly/rarely traversed documents".
- ◇ Client-initiated prefetching and server-initiated speculative service are "complementary".

# Conclusion

In a Client-Server model, servers are in a much better position to discover and utilize information about locality of reference, whether *temporal*, *spatial*, or *geographical*.

- ◇ Temporal locality of reference could be exploited to disseminate information closer to clients, to complement client-initiated caching.
- ◇ Spatial locality of reference could be exploited to initiate service speculatively, to complement client-initiated prefetching.
- ◇ Geographical locality of reference could be exploited to optimize the placement of replicas, to match the paterns of demand from clients.

For current and future projects, visit our Research Group Home Page at

`http://cs-www.bu.edu/groups/oceans`