Evaluating Server-Assisted Cache Replacement in the Web

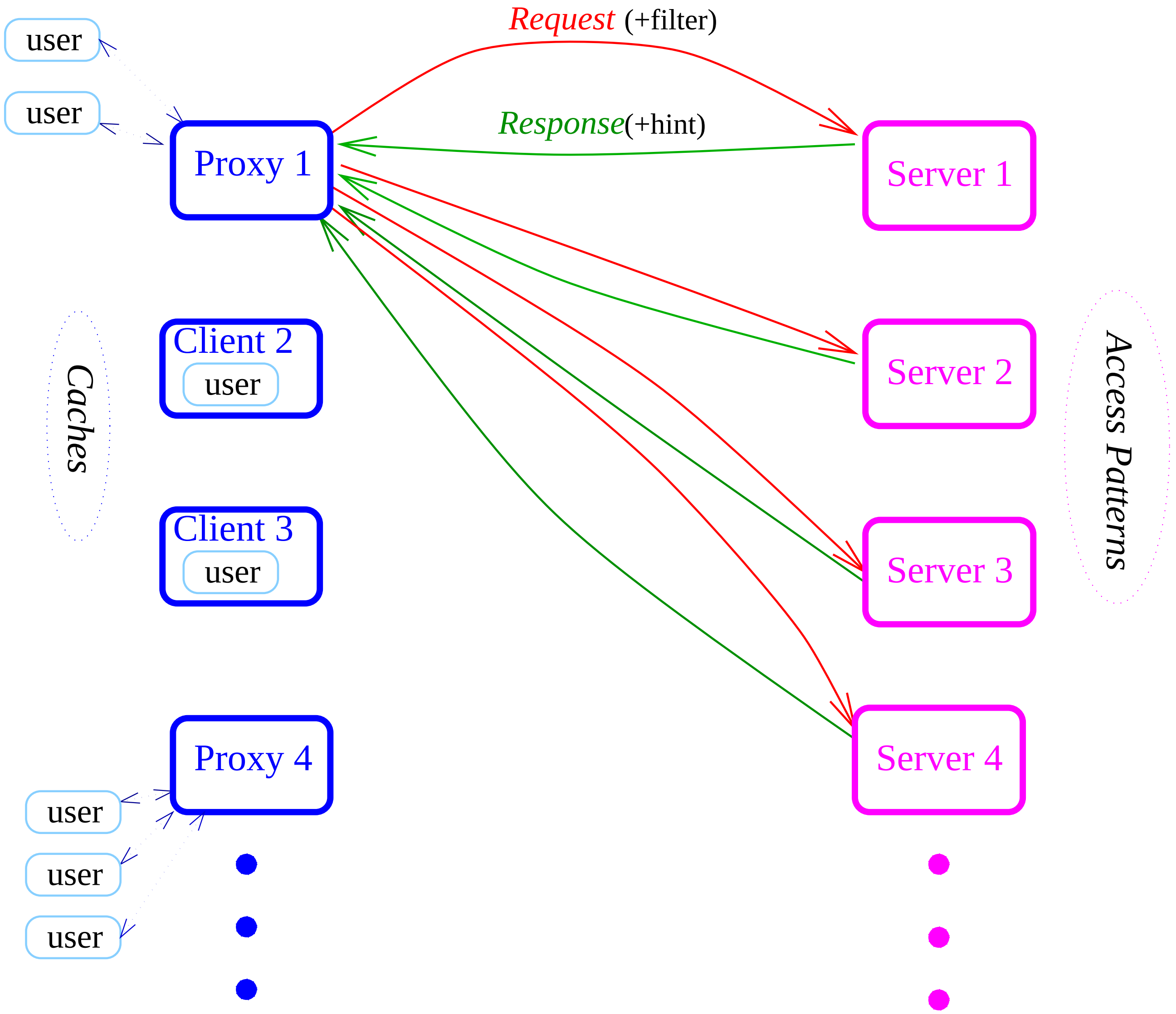
Edith Cohen
AT&T Labs – Research

Joint work with:
Balachander Krishnamurthy and Jennifer Rexford

http://www.research.att.com/∼edith

edith@research.att.com

Web Servers and Caching Clients



Improving Web Performance by Bridging Server-Client Information Gap

- What web server knows best:
  Access patterns to its resources, modification time, rate of change
- What proxy/client knows best:
  Its caching policies, staleness allowance, users behind proxy

Principles of information exchange: [MP95,B95,CKR98,...]

- ◇ No extensive per-proxy data collection at server
- ◇ No extensive per-server data collection at proxy
- ◇ No significant increase in network load or latency
- ◇ Transparent to non-participating entities

Communication Mechanism:

Information is piggybacked on normal request and response messages.
Proxy sends a filter characterizing its needs.
Server's response depends only on the contents of request and filter.

Applications: prefetching, cache management,...

Server-Assisted Cache Replacement

• Server-assisted replacement policies incorporate statistics collected at web servers to prioritize pages at the client's cache [M98,CKR98].

Mechanism: Extra information is piggybacked on the response msg:
• Distribution function of inter-request times for requested page.
• Related-pages information

• Local replacement utilizes information available locally at the clients cache.

We evaluate and compare performance of server-assisted vs. various local replacement policies.

Challenges:
•? Analytical: how to incorporate stats in replacement policy
•? Experimental: how to compare and evaluate policies at client-end using server logs.

Outline

- Incorporating access patterns in replacement policies
  - The models
  - Patterns in the Web environment
- Experimental setup for server-assisted policies
  - The challenge:
    - proxy logs do not have server access patterns
    - server logs do not contain complete workload of clients
  - Our approach: Interval-caching
- Experimental Evaluation
  - Data (server logs)
  - Replacement policies (server-assisted and local)
  - Results
  - Conclusions

Incorporating access patterns in replacement policies

- Per-page IRD (inter-request-time distribution):

Model: Each request "arrives" with a probability distribution of next request time for the page. [LPR94,KLPRS95]
We "approximate" the IRD of each page by the histogram on inter-request times collected either locally or across many clients, at the Web server.

- Exploit interdependencies between pages

Much research was done on modeling and using locality of reference for prefetching and cache replacement, by theory, operating systems, and Web researchers.
We identify pages that are likely to be requested soon after other page(s) [CKR98], and use "related pages" to enhance replacement policies: If a currently-cached page p is related to a currently-requested page, the "priority" of p is prolonged.

The case for utilizing inter-request statistics

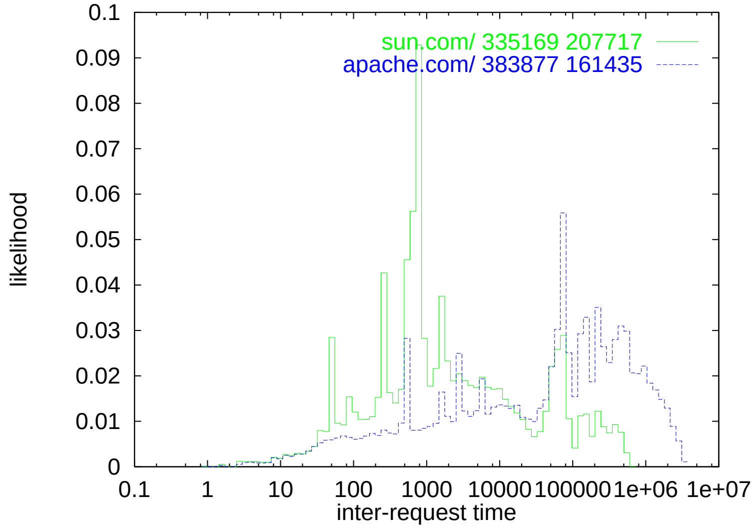
Actual inter-request-time distributions (IRD's) show that Web pages have distinct "characteristic" IRD's:
Parameters such as mean, median, and variance, as well as the shape of the d.f. curve vary considerably between web sites and within pages served by the same site.

- LRU treats all recently-requested pages equally.
  ("assumes" identical non-increasing IRD's)
- LFU prefers frequently-requested resources.
  ("assumes" fixed frequencies)

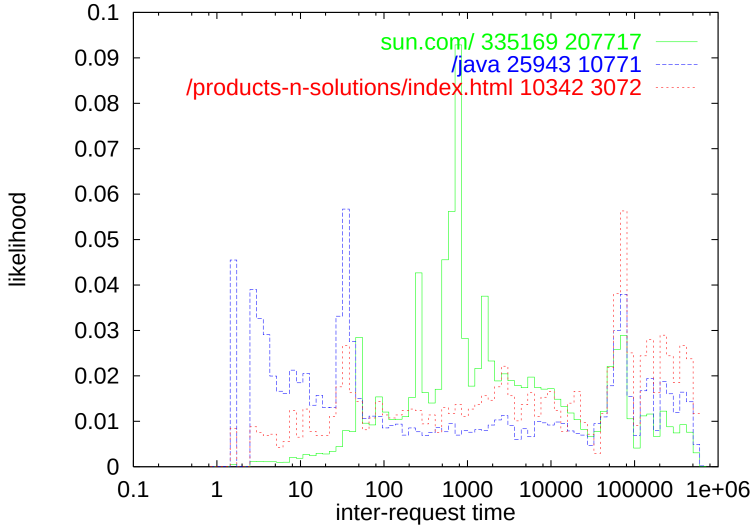
$\Rightarrow$ more refined use of the IRD can yield better results.

- Optimal use of IRD is hard [LPR94].
- We evaluate a policy which is near-optimal under reasonable conditions [CK98].

Examples of Inter-Request Distributions (IRD's)

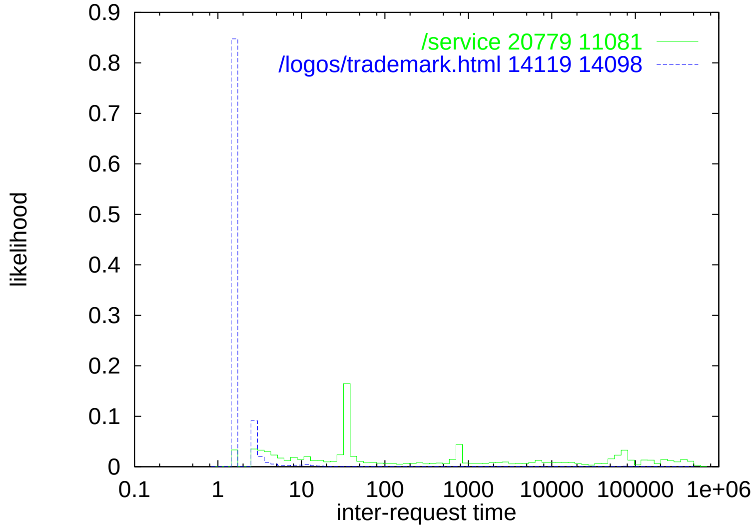


(a) Apache and Sun homepages

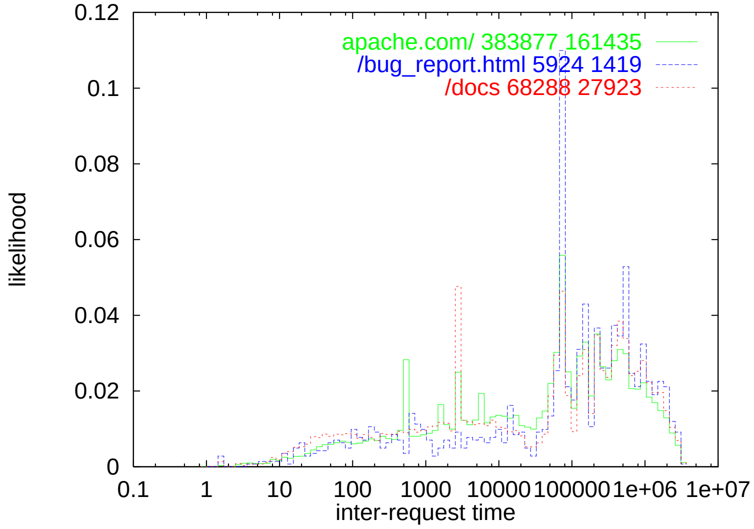


(b) Sun - various pages

More Inter-Request Distributions (IRD's)



(a) Sun pages



(b) Apache pages

Issues in the experimental setup

Proxy logs: all requests made by a client over a period of time.
Server logs: all requests made to a server over a period of time.

• Server-assisted replacement happens at the client's cache, but uses information present in many server-logs.

Ideal Data: a proxy log, and many server logs for servers contacted by the proxy, for same time period. $\Rightarrow$ not feasible experimentally.

Our approach: we use server logs and an interval-caching framework.

Server logs contain, for each client, all of its requests for the server's pages. A complete workload of each client's cache is missing, and simulating it involves too much arbitrariness.

Interval caching: we don't really need a full workload! It is enough to compare actions of different replacement policies on the subset of requests issued to our server's pages.

Interval Caching

- Complete-Workload Evaluation: Replacement decisions evict pages to fit in more profitable pages. Performance of replacement policy on a request sequence is measured by the trade-off between:

  hit rate (profit from hits) vs. cache size.

- Interval Caching Evaluation: replacement decisions determine for each requested page an expiration time. The expiration time may be extended upon new requests. We measure trade-offs between

  hit rate (profit from hits) vs. average cache portion allocated.

- Replacement policies have interval-caching counterparts.
- The complete-workload and interval-caching models are equivalent under certain assumptions.

Theoretical and experimental justification is in [CK98].

Experimental Evaluation

Our experiments:
We evaluated and compared the interval-caching counterparts of different policies on 3 Web-server logs.

- On each server log, for each replacement policy, we evaluated tradeoffs of the

  cumulative hit rate vs. cumulative "caching cost"
  (average cache sizes) across all clients.

- Replacement policies:

  - server-assisted and local versions of inter-request distribution (IRD) based policies.
  - two other local policies:

  LRU Least-Recently Used:
  EMV Exponentially-averaged Mean and Variance [Jacobson 88]

Data: Web server logs

Server logs characteristics

Log (days)	Number of Requests	Number of Clients	Requests per Source	Resources Considered
AIUSA (28)	180,324	7,627	23.64	1,102
Apache (49)	2,916,549	271,687	10.73	788
Sun (9)	13,037,895	218,518	59.66	29,436

IRD based replacement policies

Expiration times are derived with respect to a threshold parameter from

- Page fetching cost (cost of a miss)
- Page size
- Page IRD (Distribution on inter-request times)

For each page r, we use a histogram of its inter-request times for r as the IRD.

- Server-assisted replacement policies utilizes inter-request times from all clients. (SIS)
- Refined server-assisted separates clients by volume of requests and collect and disseminates statistics independently for each client class. (SIS-c)
- Local policy: only client's local inter-request times are used. (LIS)

More proxy-local policies

We evaluate two additional proxy-local policies:

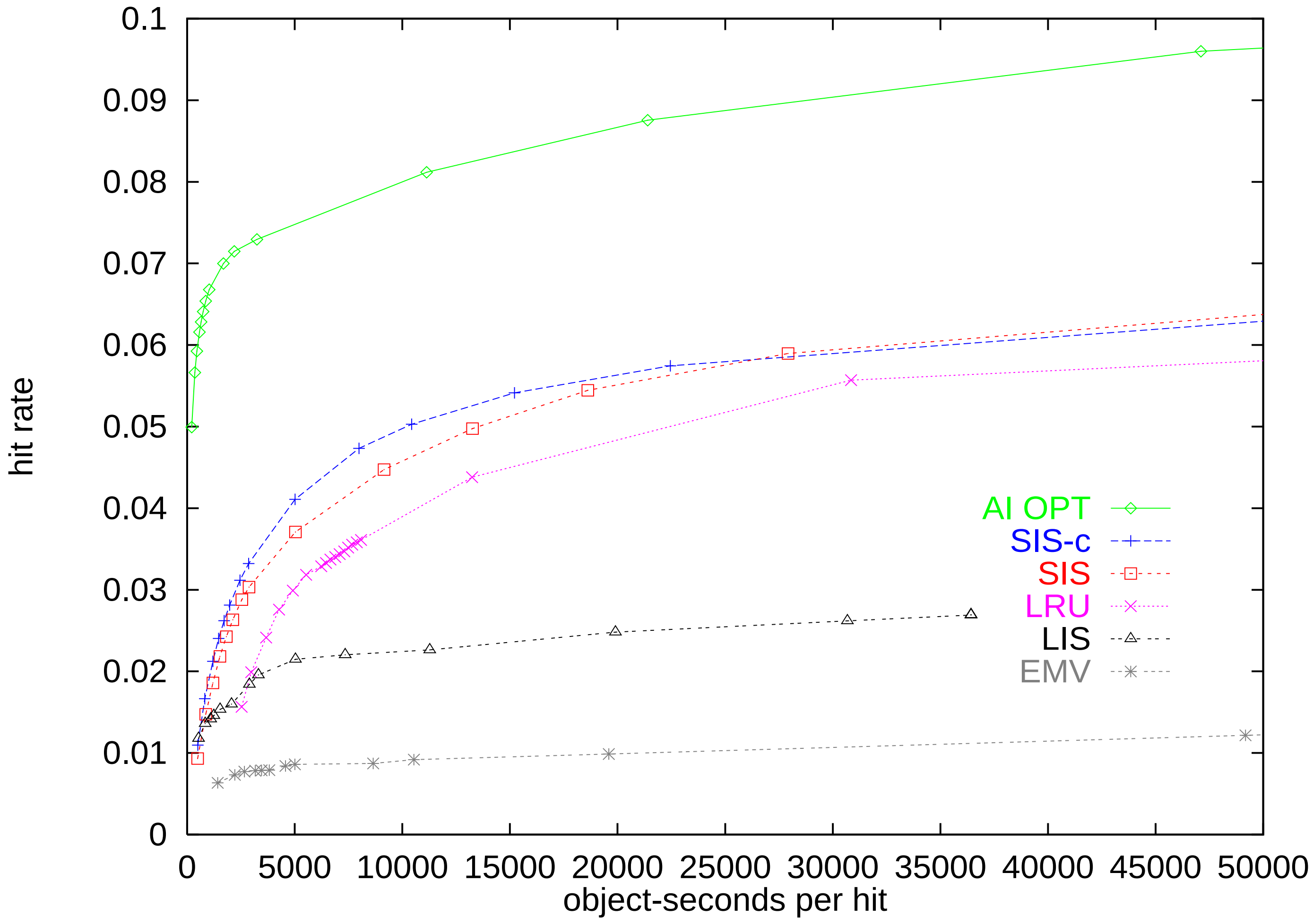
- LRU Cache all pages for same period of time T
  Trade-offs are obtained by varying T.
  For varying fetching cost f and size s use same $T * f/s$ across pages – this is the interval-caching analog of Greedy-Dual-Size [CI97,Y98].

- EMV Exponentially averaged mean and variance.
  Compute online:

$$\begin{aligned}\mu_{k+1} &= \alpha t_k + (1-\alpha)\mu_k \\ \sigma_{k+1} &= \alpha|\mu_k - t_k| + (1-\alpha)\sigma_k\end{aligned}$$

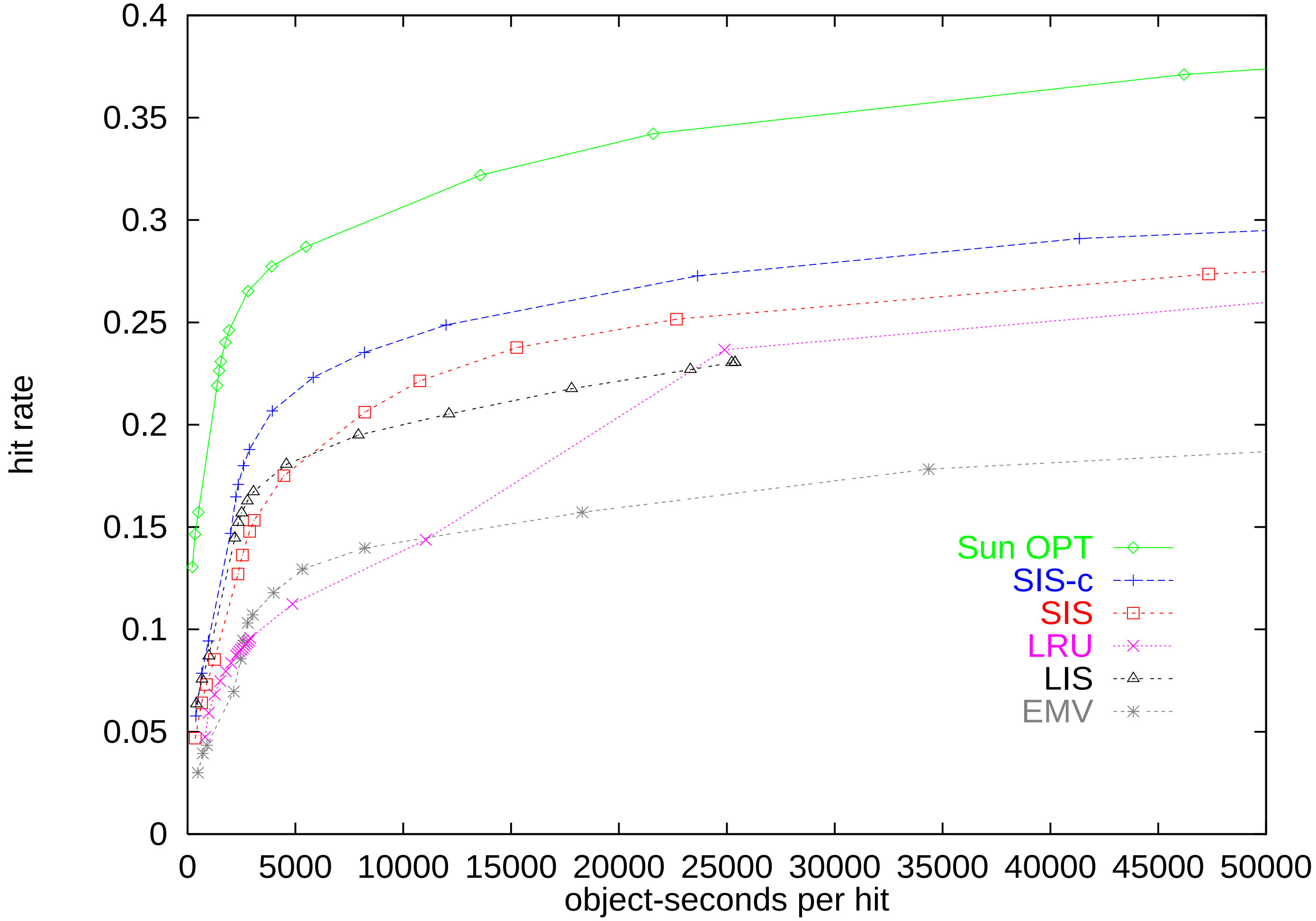
  cache for $\mu + 2\sigma$ time, if $\mu + 2\sigma \leq T$
  Trade-offs are obtained by varying T.
  For varying fetching costs and sizes, use: $\mu + 2\sigma \leq T * s/f$.

All policies, AIUSA log



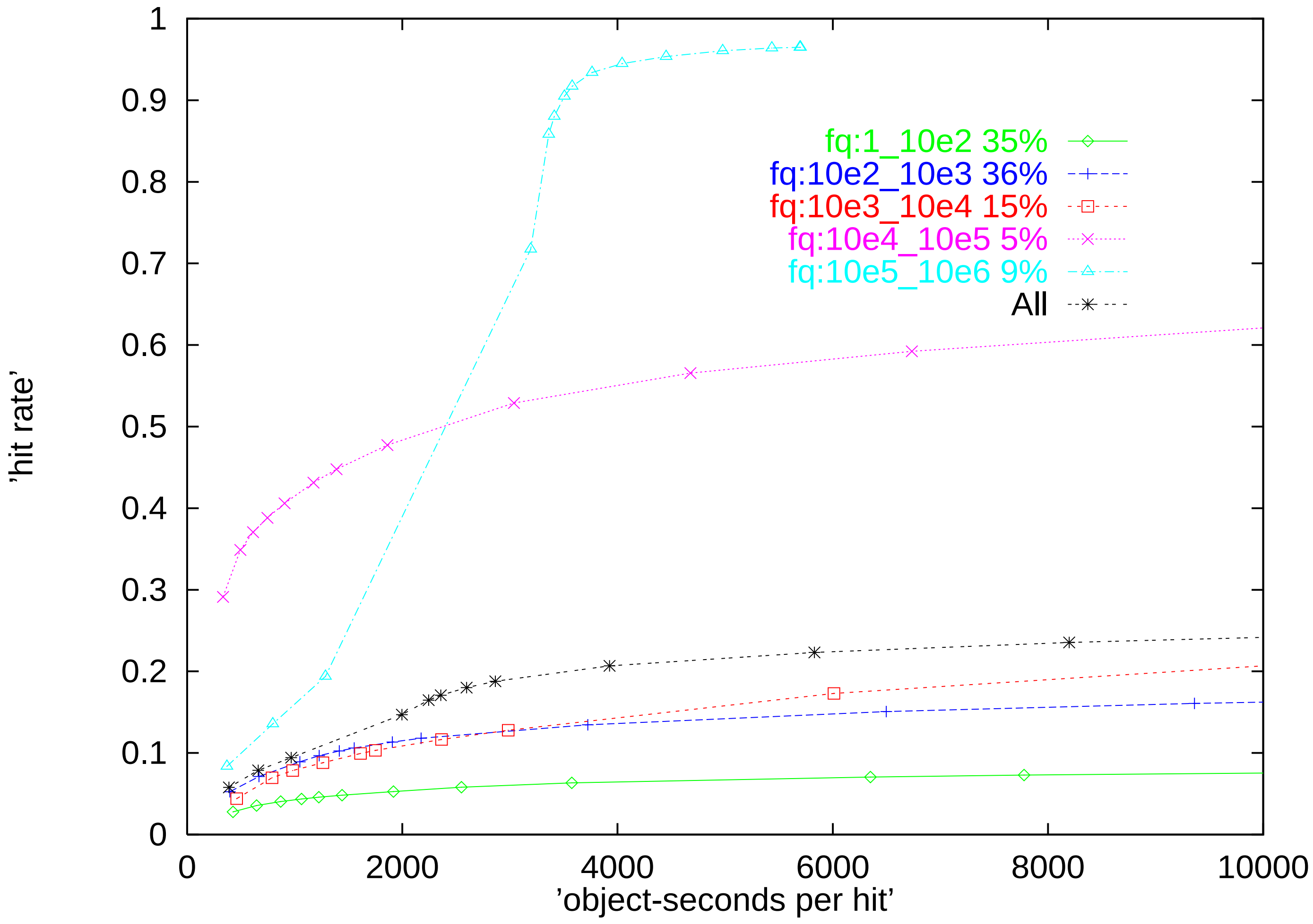
Hit rate vs. cost-per-hit for all six replacement policies

All policies, Sun log



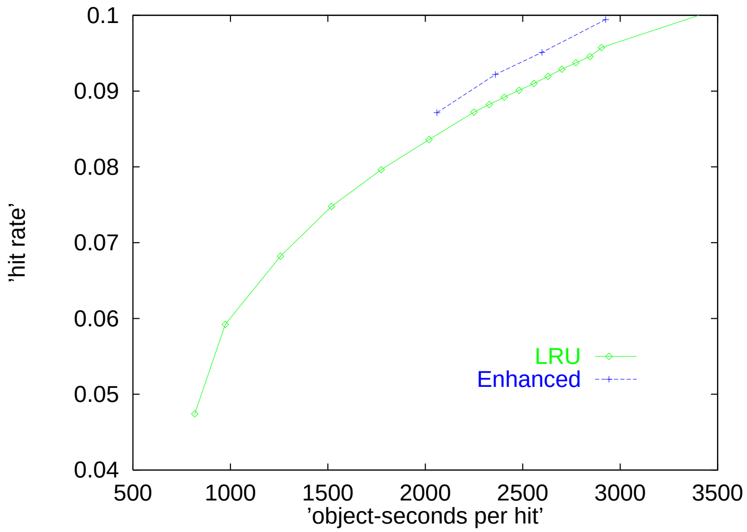
Hit rate vs. cost-per-hit for all six replacement policies

Differentiating clients by no. of requests

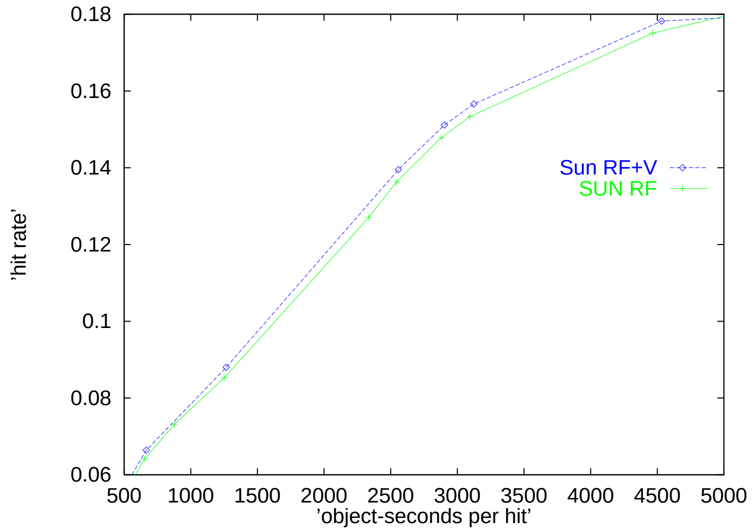


Related-pages enhanced policies on Sun log

Expiration times were extended upon requests for “related pages.” A page r_1 is related to r_2 if a request for r_1 is likely to be followed by a request for r_2.



(a) LRU vs. LRU+VOL



(b) SIS vs. SIS+VOL

Observations and Conclusions from Experiments

LIS, SIS, and SIS-c exhibit tradeoff between amount of available statistics and its applicability to a proxy client.

Observations:

- LIS performs well on the low cost-per-hit range(frequently-requested pages).
- SIS-c generally outperforms SIS and LIS.

Conclusions:

- Domination of SIS-c means that considering several client-classes is a good mid-point between local statistics and same statistics over all clients.
- For most (page,client) pairs, there is not enough statistics available locally. Hence, server-assistance is valuable. However, if there is enough statistics available locally, it is the best one to use. Suggests the adaptive policy:
  For each page: if there is enough local statistics, use LIS.
  Otherwise, use SIS-c and if this is not sufficient, use SIS.

.... Observations and Conclusions

- LRU outperformed EMV and LIS on higher cost-per-hit ranges (infrequently requested pages).
  This is due to the "technicality" that EMV and LIS perform no caching when no information was available (only 2 or 3 requests for a page by the client), but requests tend to repeat within a short time period.
  This suggests: Local policies can be enhanced by using a small default caching period.
- Hit rate is higher (for all policies) for high-volume clients.
- Gain is marginal when using related-resources information.
- Gain is large from using the IRD of each page.