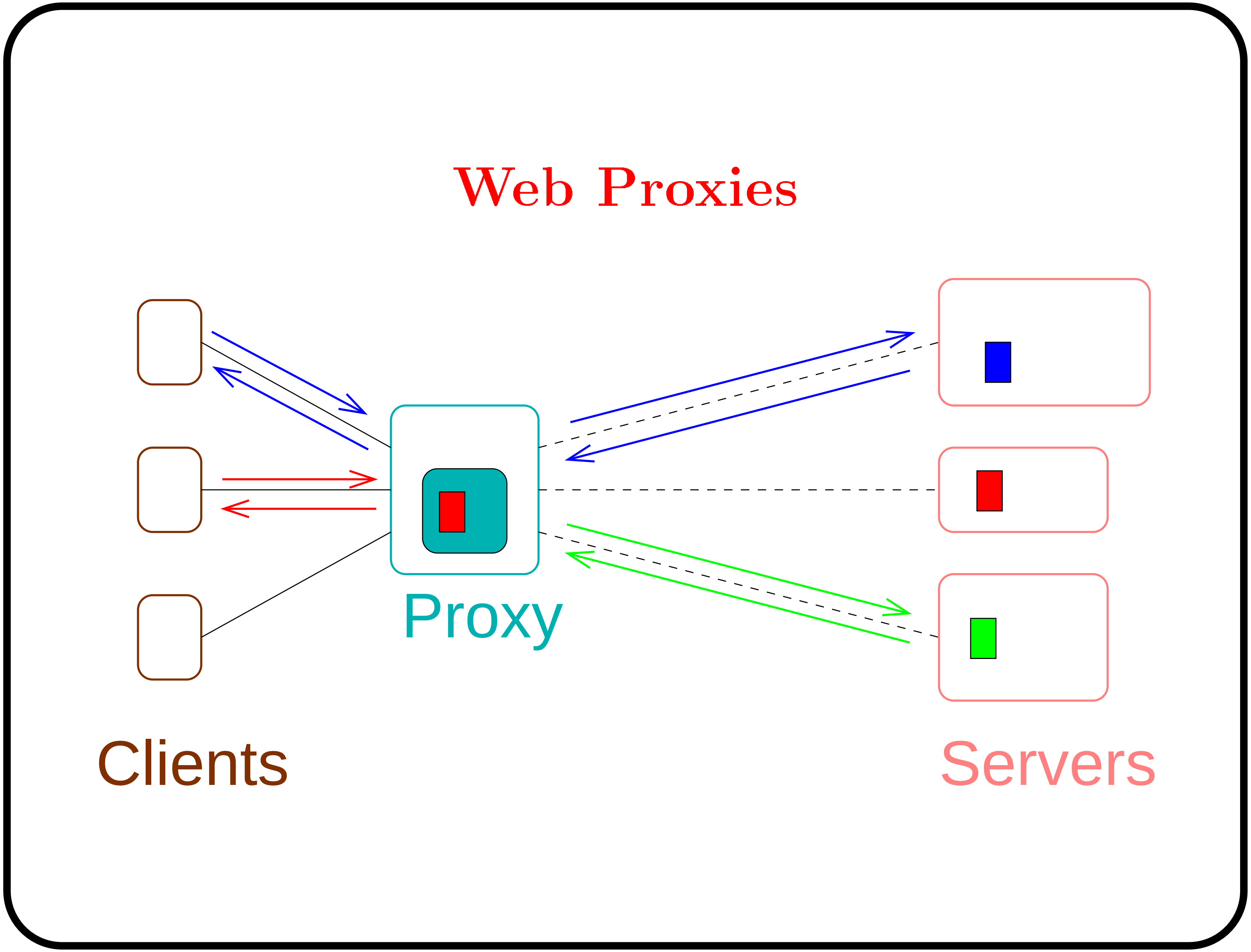
Efficient Algorithms for Predicting Requests to Web Servers

Edith Cohen, Balachander Krishnamurthy,
and Jennifer Rexford
AT&T Labs – Research; Florham Park, NJ

http://www.research.att.com/~jrex

Outline

- Motivation for proxy-server hint exchange
- Pairwise-probability algorithm
- Iterative greedy algorithm
- Sampling to reduce log processing overhead
- Performance evaluation using server logs
- Conclusions and ongoing work


Web Proxies
Proxy
Clients
Servers

Motivation for Server Hints

Proxy policies

- Cache replacement
- Cache validation
- Prefetching
- TCP connection management

$\Rightarrow$ good decisions require estimates of future accesses

Server has extensive info about access patterns

- Compute hints based on requests from many sources
- Piggyback list of hints to proxy on response messages

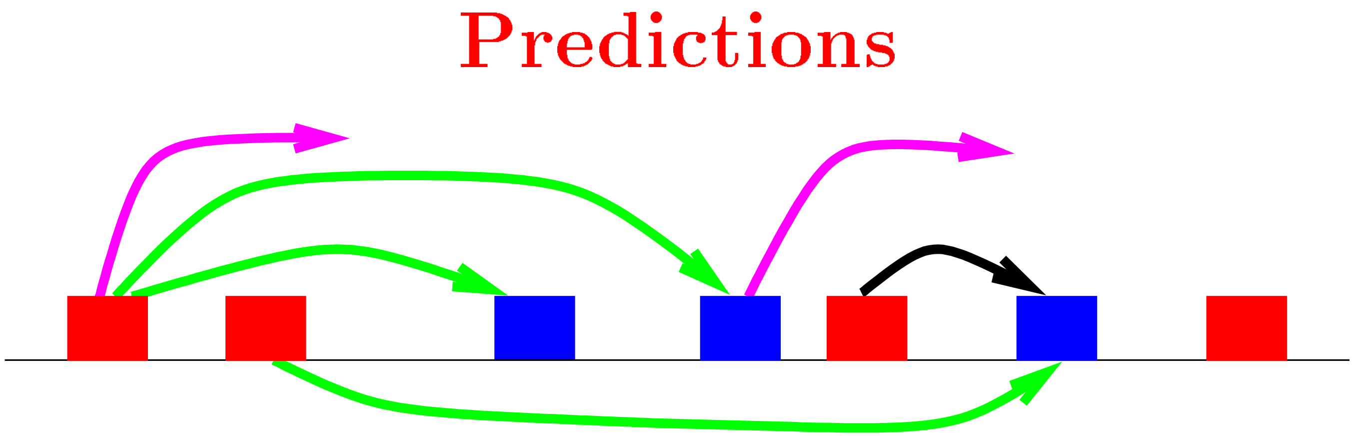
Volume Construction

Problem formulation

- **Output**: map r to a set of resources $v(r)$ (volume)
- **Input**: triple (sourceId, resource, RequestTime)
- **Evaluate**: request for r generates predictions $v(r)$

Performance metrics

- **Recall**: fraction of requests predicted by in the last T seconds
- **Precision**: fraction of "distinct" predictions that get fulfilled in next T sec
- **Hint size**: Average number of resources in prediction



- **Recall**: $3/(3+4) = 3/7$
- **Precision**: $3/(3+2) = 3/5$
- **Hint size**: $(3+3)/(3+4) = 6/7$

Volume Construction Techniques

Optimal volume-construction

- Maximize recall subject to limit on hint size
- Maximize precision subject to limit on hint size
- Problems are NP complete, so we need heuristics

Iterative greedy algorithm

- Determine resource r that generates best hints for s
- Add s to volume $v(r)$ and find next good predictor

Pairwise probabilities with thresholding

- Resources s accessed after r with high probability (e.g., probability threshold of 0.7 or 0.9)

Probability-Based Volumes

Compute pairwise probabilities from counters

- $c(r)$: number of accesses to each resource r
- $c(s|r)$: accesses to r followed by s within T seconds
- $p(s|r)$: access probability $c(s|r)/c(r)$

$\Rightarrow$ include s in $v(r)$ if probability $p(s|r)$ exceeds threshold

Example computation of $p(b|a)$

- Accesses $a, b, c, \ldots, b, d, \ldots a, d, \ldots a, e, b, \ldots a, f$
- Number of accesses: $c(a) = 4$ and $c(b) = 3$
- Counter $c(b|a) = 2$ and probability $p(b|a) = 2/4$

Several techniques for avoiding n^2 counters

Ineffective Predictions

Limitation of probability-based volumes

- Considers pairs of resources independently
- Ignores predictions generated by other volumes

$\Rightarrow$ redundant and false predictions, and larger hints

Example of ineffective prediction $p(s|r)$

- Accesses $w, x, r, s, \ldots, w, x, s, \ldots, r, t, u, \ldots, w, r, s$
- r never generates a new, true prediction for s
- r sometimes generates a false prediction for s

$\Rightarrow$ $v(w)$ should include s, and $v(r)$ should not

Reducing False Predictions

Thin volumes based on effectiveness metric

- Construct volumes using pairwise probabilities
- Measure how often predictions are both new and true
- Apply a threshold to detect poor predictors
- Remove ineffective elements from the volumes

Performance benefits

- Reduce hint size
- Increase precision
- Limited impact on recall
- Just one extra pass through the log

Motivation for Greedy Algorithm

Limitations of volume thinning

- Selecting the best effectiveness threshold is hard
- May end up thinning all predictors of a resource
- Hard to increase precision without reducing recall

Basic approach of greedy algorithm

- Iteratively add resource s to volumes v_r
- Evaluate "utility" of adding s to volume v_r
- Utility function depends on performance trade-offs
- Apply utility threshold to exclude poor predictors

Utility of Adding s to Volume v_r

- Recall: generate *new* true predictions of s
- Hint size: avoid generating a large number of hints
- Ratio: balance higher recall vs. higher hint size

$$u(s, r) = \frac{\text{number of new, true predictions of } s}{\text{number of occurrences of } r}$$

- Utility based on other volumes that already include s
- Provable performance relative to optimal volumes

Greedy Algorithm

Initialize candidate set of volumes

- Volumes v_r are initially empty for all r
- Candidate volumes d_s contain every resource

Iterate for each resource s

- For each r in d_s, evaluate the "utility" $u(s,r)$ of adding s to v_r
- Find the resource r with the max utility $u(s,r)$
- Add resource s to v_r if $u(s,r) \geq U$

Terminate when good candidates are gone

- End algorithm when $u(s,r) < U$ for all r in d_s

Smaller Candidate Sets

Reducing the size of the initial candidate set

- Candidate set initially includes all resources r
- Many resources r are not good predictors of s
- Probability-based volumes identifies good predictors
- Generate initial set d_s based on pairwise probabilities

Removing elements from the candidate set d_s

- Utility $u(s, r)$ decreases as s is added to other volumes
- Remove r from candidate set d_s as soon as $u(s, r) < U$
- Reduces computation by lowering the size of set d_s

Reducing Log Processing

Reducing the time period

- Access patterns are (usually) similar across time
- Narrow the log to a smaller time interval
- Include accesses only from this interval
- Caveats: site reorgs; time-of-day/week effects

Reducing the number of clients

- Access patterns are (usually) similar across clients
- Randomly sample a subset of the clients
- Include accesses only from selected clients
- Caveats: web crawlers, internal clients, etc.

Web Server Logs

Server Log (days)	Number of Requests	Number of Clients	Unique Resources
Sun (9)	13,037,895	218,518	29436
Apache (49)	2,916,549	271,687	788
Marimba (21)	222,393	24,103	94
AIUSA (28)	180,324	7,627	1102
EW3_4 (94)	6,273,792	79,623	2638
EW3_6 (94)	6,281,303	102,454	784
EW3_18 (94)	3,902,302	58,839	279
EW3_60 (94)	45,903,646	481,628	2,031
Nagano (1)	11,665,713	61,707	33,875

Probability-Based Volumes

Basic performance properties

- **Recall:** Predict 60–80% of accesses
- **Hint size:** Average piggyback size of 5–15
- **Precision:** True predictions 70–80% of time
- Similar for $T = 60$ seconds to $T = 5$ minutes

Removing ineffective hints

- **Recall:** Still predict 60-80% of accesses
- **Hint size:** Average piggyback size of 2–10
- **Precision:** True predictions 80–88% of time
- Best effectiveness threshold was around 10%

Performance Comparison

Greedy algorithm

- Higher recall and precision, with lower hint sizes
- Somewhat better than "best" pairwise volumes
- Very effective to use pairwise volumes as initial set
- Order-of-magnitude slower than pairwise algorithm

Log sampling

- Sampling 1/2 to 1/32 of clients is very effective
- Weekly computation of volumes is often sufficient

Conclusion

- Efficient proxy-server information exchange
- Iterative greedy algorithm
- Pairwise probabilities algorithm
- Removal of ineffective predictions
- Comparison using server logs
- Aging and sampling of logs