Summary Cache based Co-operative Proxies

Project No: 1
Group No: 21
Vijay Gabale (07305004)
Sagar Bijwe (07305023)

12th November, 2007

Abstract

Summary Cache based proxies cooperate behind a bottleneck link & serve each others requests. This reduces overhead on the bottleneck link to the servers & alleviates user latency. Earlier approaches like ICP caused considerable inter proxy messages & used computing, memory, network resources extensively.

We evaluate the performance of 'Summary based cooperating caches' by calculating number of parameters like hit ratio, false positives, false negative, network overhead & compare these with non cooperating version. We simulate proxies on different machines & use trace file to generate user requests. Our experiments & evaluations show results in favor of 'Summary based cooperating caches'.

1. Introduction

The proxies behind a common bottleneck link should cooperate & serve each other’s misses to exploit caching principle. The major benefits to be achieved using web cache sharing are reduction in web traffic and minimal user latency.

The Internet Cache Protocol (ICP) supports discovery & retrieval of documents from neighboring caches. For ICP as the number of proxies increase, the total communication & CPU processing overhead increases quadratically. A new protocol called ‘Summary Cache’ addresses the issue of scalable protocols for wide area web cache sharing.

Two critical factors that make Summary Cache exploit benefits of sharing caches are an economical representation of short summary of each neighboring proxy and the periodicity of summary update messages.

Every new scheme needs to undergo implementation & testing phases to analyze the benefits under test beds. Its experimental evaluation needs to be compared with existing schemes to address the deployment issues.

The Summary Cache protocol is implemented to study the performance of the scheme under varying loads & by tuning the various system & network dependent parameters to analyze the results. Web traffic & CPU overhead measurements for various experiments speak of the benefits & effectiveness of the sharing of caches under Summary Cache protocol. The experiments is also carried for a special case where proxies do not cooperate & the results of this set up are compared with Summary Cache evaluation & appropriate conclusions are made.

2. System Model

The system is realized be using more than two machines connected in LAN & each running at least one or more than one proxies. Each proxy runs two threads. One thread for reading the trace file, processing the request, (if request needs) querying other proxies, multi casting the update messages & other thread waiting for update messages. As updates come in, the second thread updates the summary of the proxy corresponding to the update message originator.

The second thread is constantly seeking for the update messages which causes much less latency in summary update. This is also against sequential flow where thread would be blocked seeking for update message. One more option is using timed updates but it might increase the network overhead. So this architecture increases efficiency of serving local requests & serving global update messages as well.

Each proxy is free to implement its version of bloom filter (in terms of cache size, number of hash functions, bloom filter size) & we built appropriate data structures to handle these issues. Each proxy either reads the same or different trace file but the request are similar in terms of URL (just the order is kept different, this is to ensure that proxies are requesting common queries) otherwise the trace may not entirely be inter linked.

The summaries are updated via sending the differences. We also added new packet types (besides UPDATE packets) namely Q_FN, Q_FP, A_FN, A_FP (e.g Q_FN packet sends 'url' of requested object to pertaining proxy to check for False Negative & A_FN says yes or no depending on whether it has that object or not) for calculating the false positives & negatives.

Since UPDATE messages are critical & a loss of UPDATE message (which is actually the difference), might result in indeterminate behavior, we preferred TCP over UDP for inter proxy communication.

We programmed summary caches & build the system in C language. We ran four proxies each on separate machine (to have more realistic scenario) & measured performance parameters. We also ran a version implemented with no cooperation.

3. Experimental Evaluation

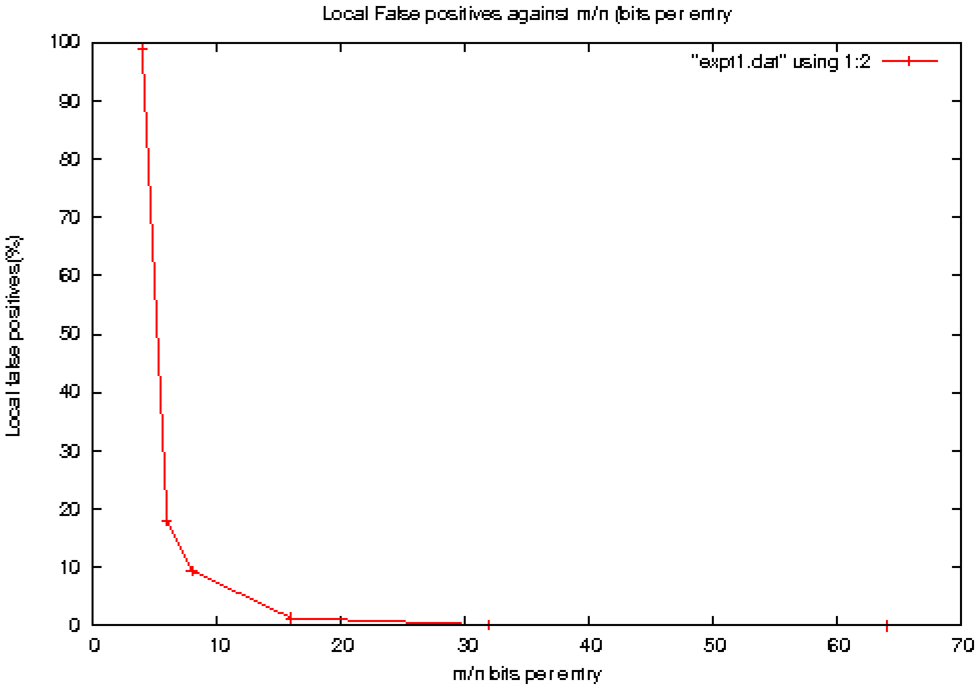
- Experiment 1

Goal: To measure number of local false positives against bloom filter size
Reason: As the bloom filter size increases the number of false positive (local) decreases. Thus by this experiment We are able to explore trade off between memory requirement and false positives.
Setup: The bloom filter size of local proxy is set as per the experimental requirements.
Parameters measured: False positives
Parameters varied: Bloom filter size

Experimental Results:

We calculated bloom filter size as a function of cache size and average file size (which gives average number of objects in cache) and number of hash functions used. For our experiments average file size was fixed to 8KB and number of hash functions to 4. It was observed that there is clear trade off between size of bloom filter and number of false positives. As size of bloom filter was increased the number of false positives decreased rapidly at earlier stages. As we increased bloom filter size further, the reduction in false positives was a bit slower. So we came up with optimal bloom filter size to be 32 bits per entry. This size is kept fixed for rest of experiments. This is evident from following graphs.

Graph:



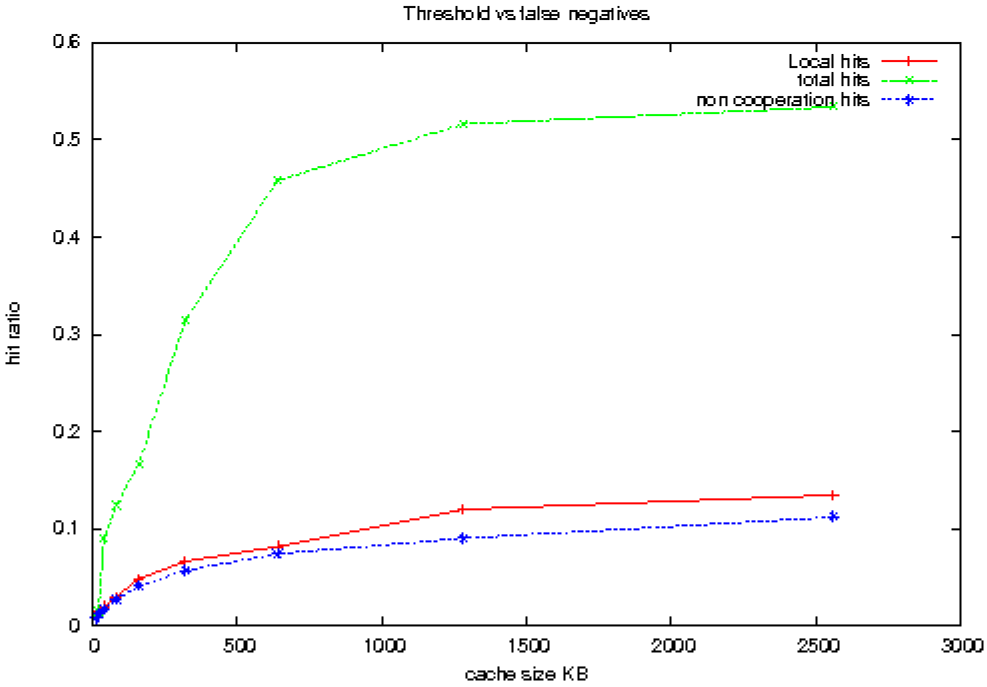
- Experiment 2

Goal: To measure hit ratio (local) against cache size
*Setup:*The cache size is varied from range 8K to 2.56 MB
Reason: The hit ratio is proportional to cache size, this experiment tests correctness of our cache implementation, blended with summary cache.
Parameters measured: hit ratio for local as well local & remote hits
Parameters varied: cache size

Experimental results:

Our caches are based on FIFO policies. As the size of cache increases the hit ratio also increases which validates our cache implementation. Local hits are the requests found in local cache where as remote cache are the requests found in neighboring proxies. Because of cooperating caches the total hit ratio, local & remote, increases considerably as we increase cache size (& in turn bloom filter size) as compared to only local hits. Following graph shows this improvement in hit ratio because of co operating summary caches. The threshold was kept 2 %.

Graph:



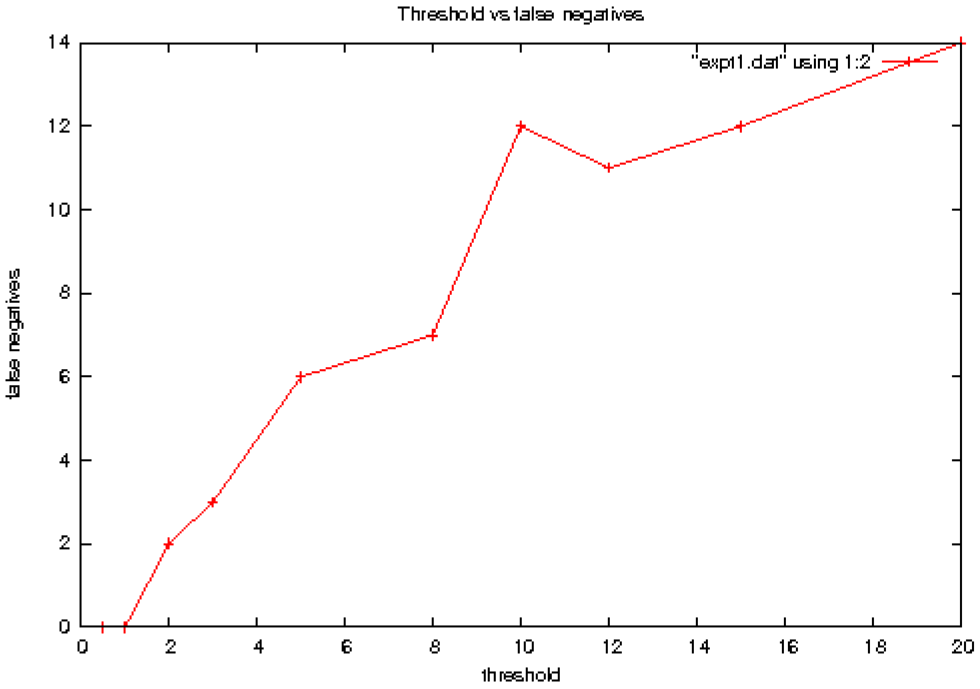
- Experiment 3

Goal: To measure number of false misses against delay of update messages
Reason: We investigated delaying the update of summaries until the certain percentage of cached documents that are new reaches a threshold. The number of false misses tends to be proportional to the number of documents that are not reflected in the summary. Hence as threshold increases number of false misses also increase. Other approach can be just sending update after specific interval of time, which we call as timed updates.
Parameters to measure: Number of false misses
Parameters to vary: Periodicity of update messages
*Metrics to be used:*Threshold(% of cache docs that are new), time(sec)

Experimental results:

Our results follow the logical *Reason* & as we delayed the update messages, the number of false positives also increased. This increase was quite linear which shows direct relationship between update interval & false negatives. We looked into neighboring proxies if we did not find object using local or remote bloom filters stored locally to calculate false negatives. Following graphs show behavior of false positives against delay in updates. The cache size was kept 1024KB.

Graph:



- Experiment 4

Goal: To measure hit ratio against periodicity of update
Reason: Experiment 3 & relationship between false misses and hit ratio indicate that this experiment is reasonable to do.
Setup: The frequency of update messages would be controlled by setting threshold for number of entries not reflected in remaining proxies or time period & hit ratio would be measured.
Parameters to measure: hit ratio
Parameters to vary: periodicity of update
Metrics to be used:Threshold (% of cache docs that are new),time(sec)

Experimental results:

As the threshold or update interval increases, the number of unreflected entries in the local remote-bloom filters increases. This slowly undermines the hit ratio as proxy was not conveyed the recent change & all it got to see was stale state of filters which was of no use for recent requests. We varied both threshold & update interval & confirmed this behavior. The cache size was kept 1024KB.

Experiment 5

Goal: To measure false positives against periodicity of update

Reason: Beyond a certain limit false positives become intolerable & this experiment helps us find out the optimal value of threshold & update interval taking false positive as constraint.

Setup: The frequency of update messages would be controlled by setting threshold for number of entries not reflected in remaining proxies or time period & false positives would be measured.

Parameters to measure: false positives

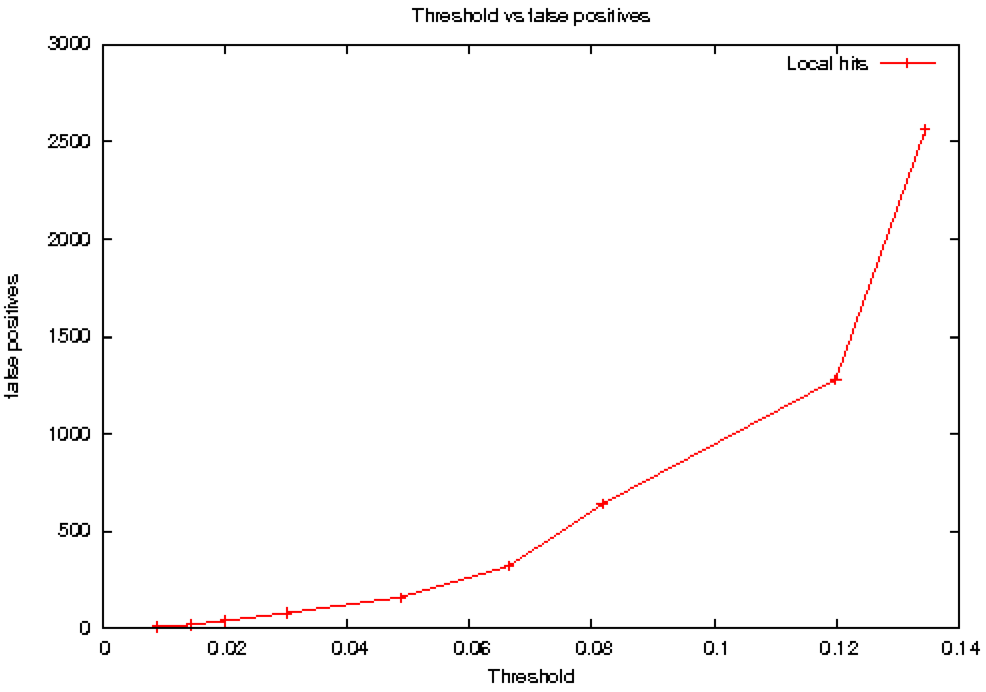
Parameters to vary: periodicity of update

Metrics to be used:Threshold (% of cache docs that are new),time(sec)

Experimental results:

As threshold increases, the number of false positives also increase as the number of unreflected entries in remote proxies increase.

Graph:

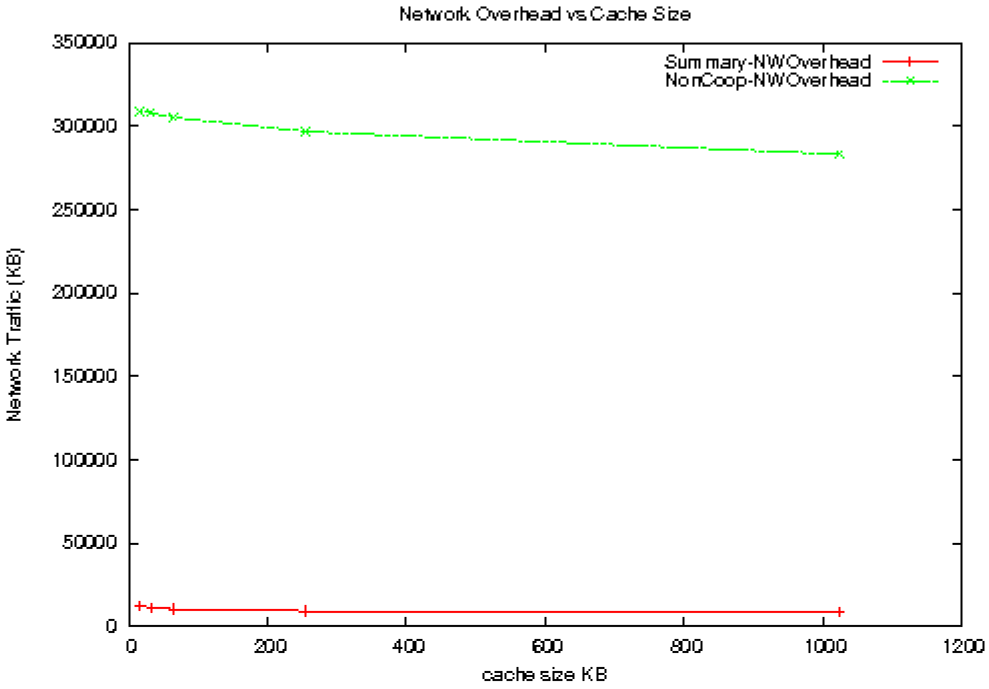


- Experiment 6

Goal: To measure network overhead against memory requirement (size of bloom filter)
Reason: The memory requirement is determined by the size of individual summaries and the number of cooperating proxies. Since the memory grows linearly with the number of proxies, it is important to keep the individual summaries small.
Parameters to measure: network overhead.
Parameters to vary: Bloom Filter Size(bits), cache size
Experimental results:

We observed that, as we increase the bloom filter size, the local network traffic of course increases. But the traffic to fetch the objects from server is considerably constrained as most of the time neighboring proxies used to have the required object. The threshold was kept 2 %.

Graph:



Comparison with non-co-operative proxies:

For each cache miss, non-co-operative proxy has to fetch the object from server which increases the network overhead of bottleneck link & also increase user latency. Also since proxies are not co-operating, they can not exploit the benefits due to sharing. This causes sharp decrease in hit ratio as compared to co-operating proxies as we increase the cache size.

Experiment 7

Calculation of CPU overhead (top tool in UNIX could be used) : The CPU utilization used to vary between 60 to 70 % which is not causing any considerable overhead.

Recommended Configurations:

Combining the above results, we recommend the following configuration for summary cache approach

Update Threshold should be between 0.5 to 2 % of Local summary

If time based update is used, then interval should be set to 1 to 2 seconds.

For bloom filter a factor between 16 to 32 works well. (Number of bits per entry)

Scalability:

The system is inherently built to be scalable. We tested the system by running 8 proxies each on different machine.

4. Usage:

The program takes switches (e.g. if user types <executable> -h, he is rendered all of the experiment switches & the parameters the program would take as HELP).

For each of the above mentioned experiments we have given a separate switch & the parameters. After the experiment is done, output is dumped in a file.

5. Conclusion

Thus, we demonstrated the benefits of web proxy cache sharing using trace driven simulations & measurements. The bloom filter based summaries with update delay thresholds, has low demand on memory and bandwidth, yet it achieves high hit ratio. As compared to non co-operation based proxies, the hit ratio is almost tripled while at the same time network overhead is reduced marginally. We also investigated the trade off between load factor of bloom filter and false positives & also between threshold of updates and hit ratio, false positives-negatives & came with optimal values for both.

6. Bibliography

Li Fan, Pei Cao, Jussara Almeida and Andrei Z. Broder (2000) Summary Cache: A scalable wide-area web cache sharing protocol. *IEEE Transactions on Networking, Vol 8, No 3, June 2000*