TERENA TF-Cache
Small project 99-005
Extended Cache Statistics

Deliverable 1 - request for peer comments

Jens-S. Vöckler 10. Mai 1999

1 Overview

The project "extended cache statistics" focuses on delivering a long term view of data obtained from a cache log file, originating from a Squid or possibly a NetCache. The purpose of this document is to outline a mandatory set of data to retrieve from the log files. Note that the document will not address problems related to the extraction of the data. The database design will be addressed in a later deliverable.

1.1 Introductory Comments

With the help of a SQL database, different views to the various performance data can be obtained. Two intervals I_1 and I_2 have to be defined during the database setup. The first interval I_1 defines a period of time for which most of the data is being aggregated, e.g. a daily period. For the peak value evaluation, a finer granularity is necessary, resulting in an interval I_2 for the peaks, e.g. an hourly view. After their initial assignment, the intervals cannot be changed.

All data should be put into the database in a form which allows for maximum precision. For instance, the percentage calculations should be based on the stored absolute values. The change into percentages will be done during the retrieval and output phase, e.g. by a matching SQL statement. Almost all the data examined needs a similar set of sums to be stored:

- Number of requests (without dimension).
- Size of the matching objects (dimension: byte).
- Amount of HITs as absolute value (requests and size, both).
- Duration (dimension: milliseconds).

Although the data will be stored in any order within the database, the output needs to match contradictory sorting criteria. For most administrators, two alternative kinds of views are believed to be of use:

1. Sorted by the number of requests, and
2. Sorted by the size of the objects transferred.

The duration mentioned above can be used to calculate a bandwidth of the transmitted data. One has to be aware that the duration from a Squid log file is a highly inaccurate value, and thus results based on calculations containing this duration in its terms should be regarded with suspicion. The value stored into the database is usually a sum of single durations. Unless mentioned otherwise, the sum is independent of HITs or MISSes.

The DFN caching project is sponsored by the German Ministry of Education, Science, Research and Technology (BMBF) through the German Research Network Association (DFN Verein). The Trans-European Research and Education Network Association (TERENA) sponsors the "Extended Cache Statistics" project.

1.2 Taxonomy

As shown in figure 1, three kinds of traffic can be observed. All queries to the cache are logged. Thus, the traffic from the *client side* matches the complete traffic as seen in the log file. The difference to the *server side* traffic is the volume saved by the cache.

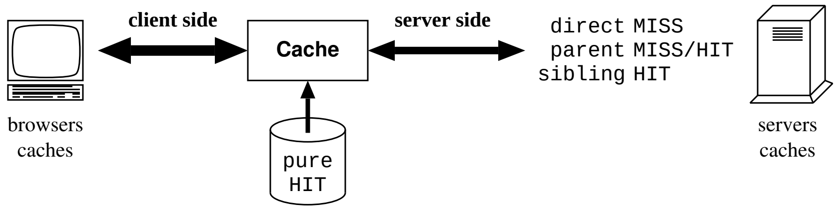

Bild 1: Traffic relations.

The well-known Calamaris-2[1] log file processor terms the *client side* traffic as "incoming". Using a different perspective, generally speaking the direction of the data stream can also be viewed as flowing from the source, the web server, to the sink, the browser. Hence, this point of view would call the traffic leaving the cache, in a sense "outgoing". In order to reduce misunderstandings, the neutral phrase *client side* will be used in this document.

In order to avoid further misunderstandings, a few more phrases are defined here. For instance, the client querying this cache can be either a browser or another cache. Dependent caches are sometimes called *children*.

The *server side* see only that part of the traffic which cannot be satisfied by the cache. Originating at the server side, queries can be sent to either other caches or the origin site. When talking about cache hierarchies, the other caches can either be on the same hierarchy level as the current cache, or they are on a higher level upward in the hierarchy. Both kind of caches are called *neighbours*[2] or *peers*. In order to distinguish the hierarchical level of the queried cache, caches on the same level of the hierarchy are called *siblings*. Caches on a higher level are called *parents*.

By querying a sibling cache, usually only HITs can be fetched.[3] If a parent cache is queried, either a HIT or a MISS can result. Finally, if going directly to the origin site, usually a MISS will be seen, unless when doing an *If-Modified-Since* query which was answered by a *304 Not modified.*[4]

Due to the frequent changes of the names used for different HITs and for the hierarchy codes, the definition of both should be kept variable, preferably part of a configuration file. Thus it is hoped to maintain a certain degree of upward compatibility and flexibility in order to cope with future changes. A variable definition should also simplify the parsing of foreign log file formats. Erroneous requests should be recognized by their HTTP status code, as Squid-2 does not log the special ERR code any longer. The scheme of keeping the definitions variable

1. http://calamaris.cord.de/ (May 1999).
2. Squid calls it *neighbor,* using the AE spelling.
3. A *false HIT* as seen when using *cache digests* cannot be detected by looking at just one log file, because an appropriate entry in the access log file is still missing (Squid 2.2s2). Currently, one would have to correlate all log files of a peering group in order to find false HITs.
4. There exist DIRECT HITs in my log files which are not refreshes. Possibly a squid bug?

can also be used to accommodate particular needs in the definition of "what is a HIT" which may vary from admin to admin.

Although Squid-1 is not supported any longer, it still has a considerable user base. Therefore a certain degree of downward compatibility seems desirable. Default will be Squid-2, though.

2 Mandatory sections

Of importance are all number which enable the administrator to look at a related group of caches as a whole. The report should account for the amount of traffic flowing into that group, going out of the group and the amount for which the group acts as a generating source. Also, the inter cache communication may eat some bandwidth, and needs to be looked at. On the other hand, an administrator of a dependent cache on a lower level of the hierarchy might be more interested in the number how much bandwidth or latency a cache saves him. The sections shown in this chapter constitute minimum requirements, which are hoped to match both points of view.

Many times the server side traffic will be view as three different kinds of traffic:

1. traffic going *directly* to the source,
2. traffic travelling via *parent* cache, and
3. traffic using *sibling* caches.

2.1 Peak values

The peak values are maintained in order to give a very coarse overview of the general cache behaviour. The classical Calamaris-1 approach for finding peak values is <u>not</u> meant.

- Separation between TCP and UDP traffic. For reasons of backward compatibility the ERR traffic should be counted as is seen fit.
  - Distinction between object size and request sum.
  - With TCP, a division between HIT, MISS and OTHER.

The results are aimed at generating a bar chart where each bar is made up of the pieces for HITs, MISSes and OTHER traffic. The diagrams generated are intended for our "pointy-haired" bosses, in a way, an executive summary. The distinction between size and requests as seen in Calamaris-2, as well as the further distinction between direct, parent and sibling traffic seems desirable.

2.2 Domains

This section focuses on TCP traffic. Many cache admins express an interest into the domains for which their cache is queried. From the 2nd level domains the top-level domains (TLD) could possibly be generated using appropriate SQL statements. A SQL view could also be a solution. We are aware of the problem that domains are very diversified, and bound to change a lot, but a solution is not part of this document. Just the requirement for a examining the domains is stated.

The possibilities of <unresolved> and <error> statements can be met in extreme situations. The first is true for numerical addresses which cannot be resolved into a domain at the time of processing the log file. The latter will be chanced upon for erroneous or otherwise unparsable URLs. Since URLs are basically user input, they are prone to errors, and should be regarded with suspicion.

2.3 MIME types

This section focuses on the client side TCP traffic. A sorting of the output is done by the rules laid out in the introduction. As in the previous section, we are aware of the problem that the key data is bound to change a lot.

2.4 Request method

There are different methods to access an object on a web server, as laid out in the respective standards. Squid-2 at the time of this writing knows about the following methods: GET, HEAD, POST, PUT, PURGE, DELETE, TRACE, OPTIONS, CONNECT and ICP_QUERY. The latter is used for inter cache communication using the ICP protocol. Most of the methods are non cachable. Unknown methods should be logged as <other>.

2.5 Overview of the requests (*client side*)

This section should answer the question "who asks me". It will look at the complete traffic in a coarse overview, sorted by the HIT or MISS status. For both protocols, TCP and UDP, a distinction into HIT, MISS and ERR seems sensible. Mind that the querying instances can be browsers as well as caches.

2.6 Generated traffic (*server side*)

This section focuses on the question "whom do I ask". It will look at the TCP traffic generated by this cache. Usually, caches at the top level of a hierarchy go to the source in a direct fashion. A certain percentage of the queries can also be answered with the help of peers. Thus, a distinction between DIRECT, PARENT and SIBLING traffic is necessary.

Additionally, for each peer the way an object was retrieved from the peer must be distinguished. Usually, an object will be found on a peer employing an ICP query, and transferring the object via HTTP. As new inter cache communication protocols arise, peer objects can also be found with the help of those protocols, e.g. cache digests. Further protocols like the cache array routing protocol (CARP) are also supported. Due to the variety of inter cache communication protocols the generated traffic issue must be kept versatile in order to match future methods of inter cache communication. Also note that the traffic generated by inter cache communication is usually only recorded in the log files of the peering caches, not the log file of the retrieving cache.

Two different views are possible. One is sorted by the hierarchy code as returned by Squid. The other view additionally prints statistics for all peers queried. Essentially, the same basic table is being referred to.

2.7 Detailed information on querying instances (*client side*)

In analogy to Calamaris-1, the traffic generated by each querying instance should be separately listed. It might be possible to combine this view with the one mentioned in section 2.5. Of interest is the TCP and UDP traffic separately, and the TCP traffic further distinguished into HIT, MISS and OTHER. Again, the problem of vastly growing tables is noted, but not addressed.

3 Extensions

Whereas section 2 listed the mandatory minimum requirements, this section shows those features which are "nice to have".

3.1 Destination *Autonomous System (AS)*

When going directly to the origin site, it might be of interest which border gateways of the network were used to what amount. The destination AS number can in turn be resolved into the border gateway address with the help of routing protocols outside the scope of this project. The yield is the amount of traffic on the external links of the network.

In order to determine the AS number, the host name of the destination first must be resolved into a numerical address. When converting the host address into a class C network address, the network address can be resolved with the help of a local `whois` mirror and thus turned into an AS number.

This section is meant to demonstrate the flexibility of the parsing software and should show the ease of extending the parser for future requirements.

3.2 Distribution of object size and request duration

Looking at TCP, for the three classes HIT, MISS and OTHER the object size and request duration can be summed up using fixed sized classes. The classes are calculated using the logarithm to the base of 2, where the zero is a valid input, and thus needs special treatment. There can be three kinds of diagrams be generated from this statistical distribution:

1. Number of objects over a size distribution (two dimensions),
2. Number of objects over a time distribution (two dimensions) and
3. Number of objects over a combined size (x) and time (y) distribution (three dim.).

The problems arising from inaccurate duration as seen in the log file are noted.

3.3 pure HITs

In sections 2.5 and 2.7 a distinct count for the NONEs hierarchy code when processing a HIT are of interest, because these type of HITs are pure, that is, they generate no extra traffic outside the cache. Otherwise even a HEAD request contributes to a higher latency, and generates traffic in the order of at least four up to eleven TCP segments.

3.4 Protocols

Of interest for all TCP-based queries is the gatewayed protocol as seen in the URL fed to the cache. The protocol is the first part of the URL before the "://", as defined in RFC 1738.

Like the tables in sections 2.2 and 2.3, it is possible for this table to grow without bounds. A possible solution might be to limit the number of protocols parsed, and record all valid names into the configuration file. All otherwise unknown protocols are counted as <unknown>. As a standard the protocols telnet, ftp, http, https, cache_object, news, nttp, wais and gopher should be part of distinct sums.