Calculating Web Cache Hit Ratios

March, 2000
Leland R. Beaumont
Lee@content-networking.com

Abstract: Web Caching can provide significant benefits to both the end user and the service provider when the requested object is served from the cache. This paper calculates upper bounds to the hit ratio based on several factors, including the number of objects available on the Internet, the size of the cache object store, the average size of an object, the expiration time of an object, the fraction of objects that can be safely cached and the popularity distribution of objects on the Internet.

Introduction

Web Caching, the caching of objects on the Internet, can provide significant benefits to both the end user and the service provider [1][2][3] These benefits are only available when the requested object is served from the cache. The ratio of objects served from a cache to the total number of requests is called the *hit ratio*. This paper calculates upper bounds to the hit ratio based on several factors, including the number of objects available on the Internet, the size of the cache object store, the average size of an object, the expiration time of an object, the fraction of objects that can be safely cached and the popularity distribution of objects on the Internet

Cache storage size

The larger the storage space the cache has, the more objects it can store and the more likely it is that a request will result in a hit. Storage size in measured in Gigabytes (GB) where one GB = 1024 × 1024 × 1024 bytes of information. This paper considers cache object storage ranging from 16 GB to 192 GB in size.

Object Size

The smaller an object is, the more objects can be stored in a given cache. Object size is measured in kilobytes where one KB = 1024 bytes. The number of objects that can be stored for a given cache size is shown in the table below. Objects of 3, 5 and 10 KB average size are considered. The average object size on the Internet is not accurately known, but is generally considered to be in this range[4]

cache Size	Average Object Size		
	3 KB	5 KB	10 KB
16 GB	5,592,405	3,355,443	1,677,722
32 GB	11,184,811	6,710,886	3,355,443
48 GB	16,777,216	10,066,330	5,033,165
64 GB	22,369,621	13,421,773	6,710,886
80 GB	27,962,027	16,777,216	8,388,608
96 GB	33,554,432	20,132,659	10,066,330
112 GB	39,146,837	23,488,102	11,744,051
128 GB	44,739,243	26,843,546	13,421,773
144 GB	50,331,648	30,198,989	15,099,494
160 GB	55,924,053	33,554,432	16,777,216
174GB	60,817,408	36,490,445	18,245,222
192GB	67,108,864	40,265,318	20,132,659

Total number of objects stored.

For example, a 32 GB cache can store a total of 11,184,811 objects whose average size is 3 KB.

Replacement Policy

Eventually the cache object store will be filled and a request will occur for a new object for which there is not room in the cache. Assuming a 32 GB cache with an average object size of 5 KB, then a maximum of 6,710,886 objects can be stored. At a request rate of 500 objects per second, 6,710,886 object requests are made in 13421 seconds or 3.73 hours. However, not all of these requests are distinct. Based on the analysis shown in Appendix B [this is the Zipf program that calculates distinct requests] approximately 40% of the requests are distinct. This means it will take the cache approximately 3.73 / .4 or 9.2 hours to fill up the cache.

Once the cache is filled, a decision has to be made on how to replace the cache contents with objects that are more likely to be useful. Three such replacement policies have been studied. They are Least Recently Used (LRU), Largest file first (LFF),and First In First Out (FIFO). Additional important replacement policies include Next to Expire (NTE) and GD_Size[5]

The LRU policy considers when the object was last read, regardless of when it was written. Under this policy the cache will eventually fill with the most frequently requested objects.

The LFF strategy replaces the largest object without considering either the read time or the write time. Under this policy the cache will eventually fill with the smallest objects.

The FIFO policy considers when the object was written. Under this policy, the cache will fill with the objects that have been most recently refreshed.

The NTE policy replaces the object which is next scheduled to expire. Under this policy, the cache will eventually fill with the most stable (infrequently changed) objects.

[research GD_Size]

Expiration Time

It is important the a cache deliver up to date information, even though objects on the Internet change. Some object, like stock quotes, news headlines and weather reports, change often and are not suitable for caching. Others, like daily news letters, quote of the day, and status reports change approximately once per day and can be cached if attention is paid to their expiration times. Many other objects, such as images and published reports, do not change, even over a period of years. These are excellent candidates for caching.

The Hyper Text Transfer Protocol [6] provides at least two mechanisms that help a cache to determine the likely expiration time for an object. The first is the "expires" field of the header. The server can provide a specific date and time after which this object is considered stale and needs to be refreshed from the origin server. The second is the "last modified time" field. This is the time when the object was either created or was lasted changed. If the object has been unchanged for a long time, it is reasonable to assume it will not be changing in the very near future.

Consider a cache with an object store of 32 GB, and average object size of 5 KB. This will store 6,710,886 objects. If this cache serves 500 requests per second, how often is the least referenced object requested? Assuming a Zipf distribution (see appendix A) and an Internet universe of 1,000,000,000 objects the least referenced object has a probability of being requested of $p(6{,}710{,}886) = 1/(6{,}710{,}886 \times \ln(1{,}000{,}000{,}000) = 7.19 \times 10^{-9}$ At a rate of 500 requests per second, the probability of requesting this object in any given second is $500 \times 7.19 \times 10^{-9}$ or 3.6×10^{-6} The expected inter request interval is then $1/p = 278143$ seconds or 77.3 hours or 3.2 days.

Considering the range of expiration times on the Internet, this object may or may not have changed since it was last requested more than 3 days ago. Under a FIFO replacement policy the cache will eventually displace objects with short expiration times. This minimizes the effect of expiration time on cache hit ratio. Under the LRU or LFF policies, some fraction of requests will be for stale objects.

[look for a reference to estimate what fraction of requests are for "stale objects" Can we get this from analyzing the exit 109 logs? What are they called? Refresh hit?]

Size of the Internet

The precise number of objects available on the Internet is not precisely known. Estimates range from 100 million to more than a billion. [get references for this]. This paper considers estimates of 10 million, 100 million and 1 billion objects. It is shown that this number is very important for estimating hit ratios if a uniform popularity model is assumed, but becomes relatively unimportant if a Zipf distribution popularity model is assumed.

Cacheability Percentage

Internet protocols provide a variety of mechanisms to allow objects to be marked as "not to be cached". These include immediate expiration times, use of the cache-control header options, use of "cookies" and dynamic pages, including cgi-bin requests.

Estimates of the fraction of Internet objects that are non-cacheable vary. [get estimates and references]. If a non-cacheable object is requested, it cannot result in a cache hit. The non-cacheable percentage directly affects the cache hit ratio observed according to the following formula.

$$p(\text{hit}) = p(\text{ object is cacheable }) \times p(\text{object is in the cache})$$

This paper considers values of 70%, 75%, 80% and 85% for caceability percentage. [need to add this in at the and of the report]

Popularity distribution

Several attempts have been made to characterize the reference locality on the WWW.[7] The popularity models considered in this paper are the Uniform distribution and the Zipf distributions (see Appendix A) with various exponents or α parameters.

Assuming a uniform distribution popularity model, any object is equally likely to be read. If a cache can hold 6 million objects, the probability of a requested page being in the cache for various assumptions about the number of objects in the Internet, is as follows:

Internet Size	Probability of Being in cache
10,000,000	60.00%
100,000,000	6.00%
1,000,000,000	0.60%

Here we see that for a large Internet (e.g. more than 100 million objects, the maximum achievable cache hit ratio is quite low, below 6%.

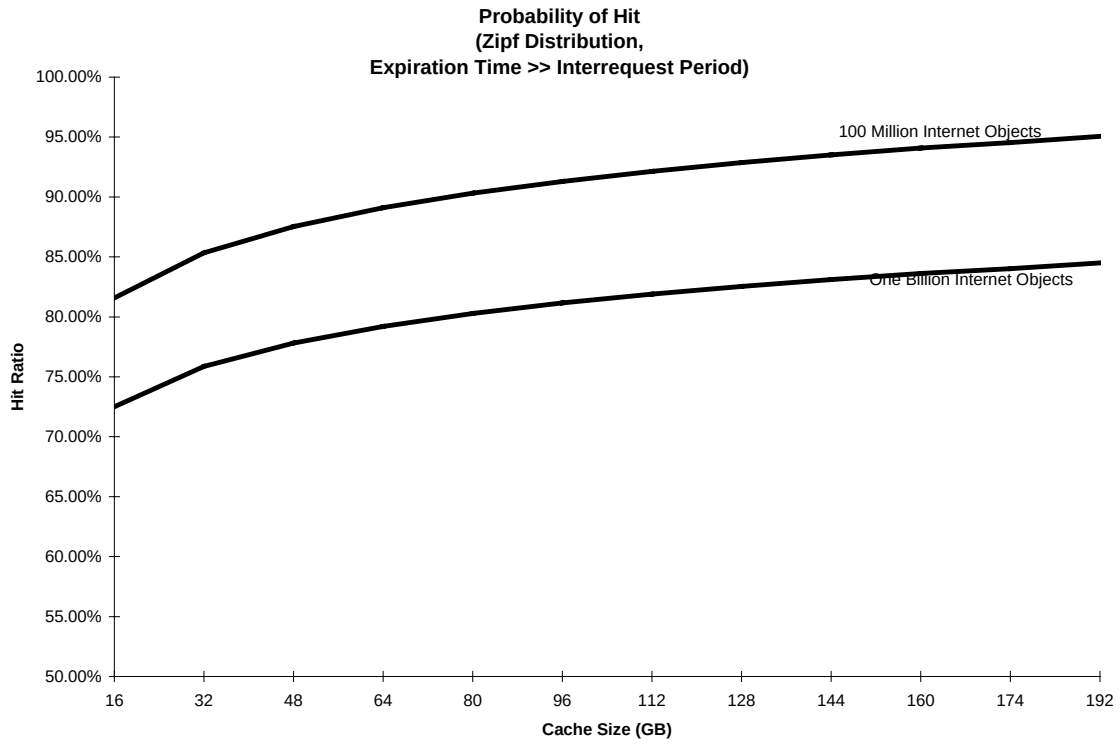
The uniform model is generally considered an inaccurate representation of Internet request popularity because it provides for too broad a locality of reference.

Assuming a Zipf distribution, the probability of an object being requested is proportional to the rank of that object. If a cache can store 6 million objects the probability of a requested page being in the cache is simply the cumulative distribution function value evaluated at 6 million. For such a cache, the probability of a requested page being in the cache for various assumptions about the number of objects in the Internet, is as follows:

Internet Size	Probability of Being in cache
10,000,000	96.8%
100,000,000	84.7%
1,000,000,000	75.3%

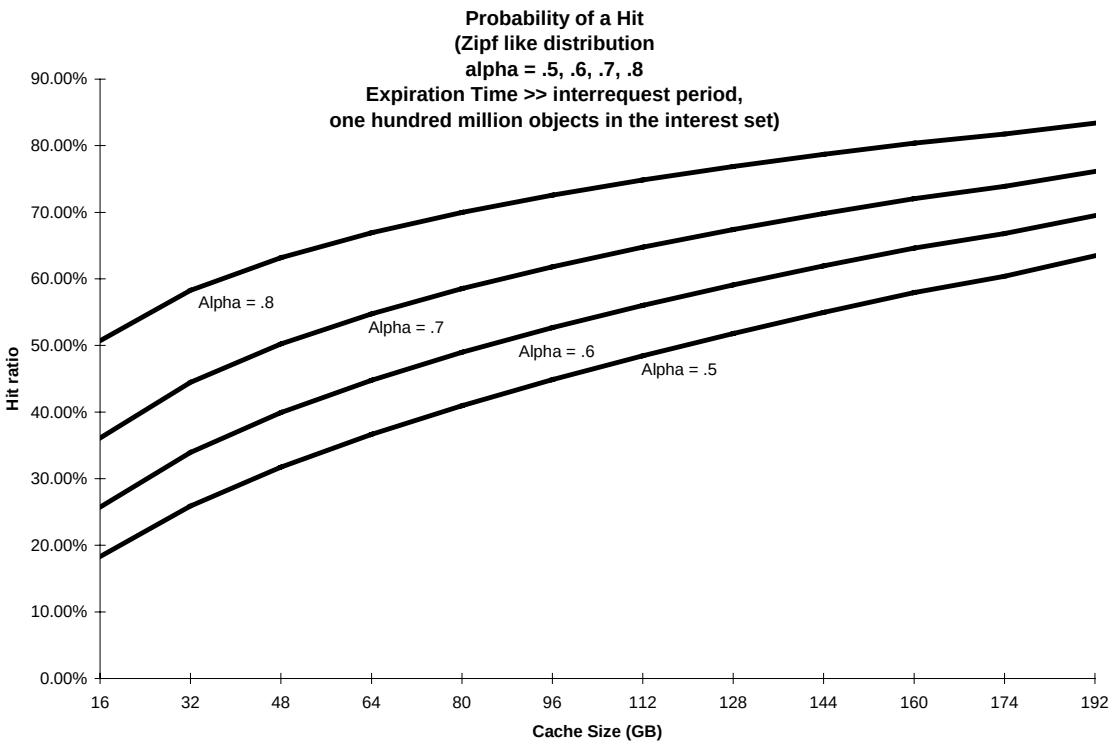
Note that the size of the Internet has a relatively small effect on hit ratio, if a large Internet (e.g. at least 100 million objects) is assumed.

Using this Zipf popularity model ($\alpha = 1$), hit ratios as a function of cache size for 5KB objects are graphed below for both 100 million and 1 billion Internet objects. This graph ignores the effect of expiration times and Cacheability Percentage.



The Zipf distribution with an exponent of one is generally considered to provide too narrow a locality of reference to be an accurate model for Internet object popularity. References[8] suggest a choice of exponent in the range or .5 - .7.

Using this Zipf-like popularity model, hit ratios as a function of cache size for 5KB objects are graphed below for both 100 million Internet objects. This graph ignores the effect of expiration times and Cacheability Percentage.


Probability of a Hit
(Zipf like distribution
alpha = .5, .6, .7, .8
Expiration Time >> interrequest period,
one hundred million objects in the interest set)
Hit ratio
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
Alpha = .8
Alpha = .7
Alpha = .6
Alpha = .5
16
32
48
64
80
96
112
128
144
160
174
192
Cache Size (GB)

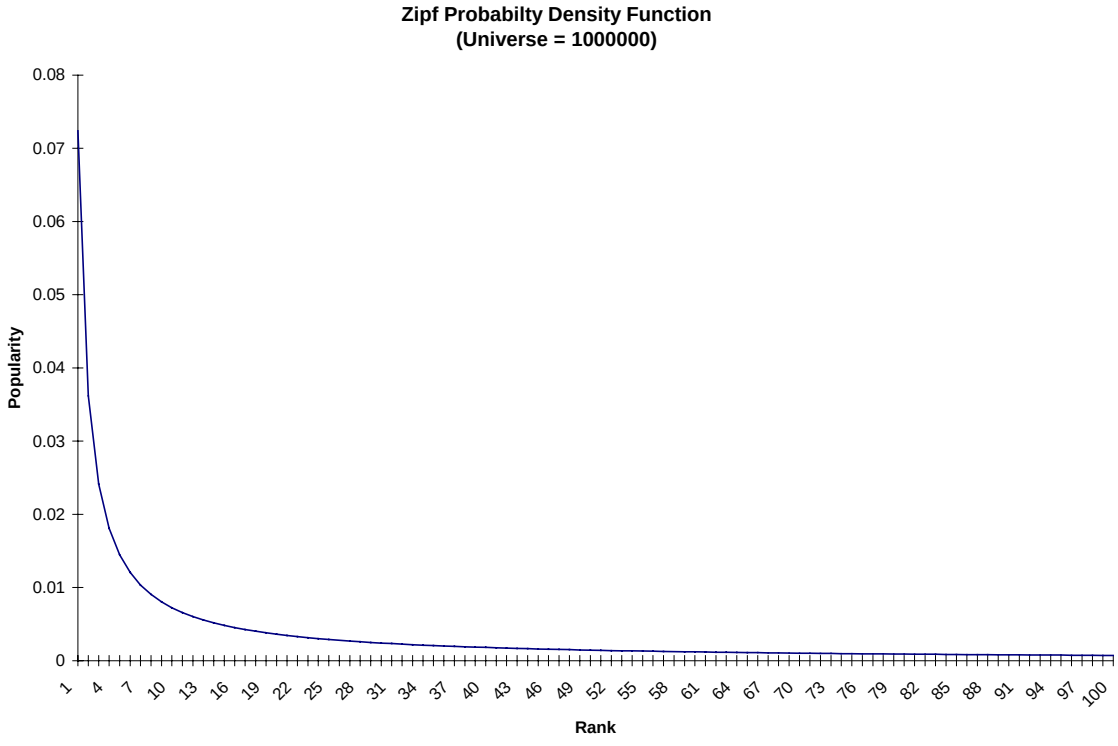
Appendix A - Zipf Distribution

Several references hold that the popularity model of objects on the Internet follow the Zipf distribution.[9,10] A Zipf distribution is one where the probability of selecting the *i'th* item is proportional to 1/*i*. Zipf distributions are empirically associated with situations where there are many equal-cost alternatives, such as choosing a book from a library or the use of words in the English language. Zipf's law, named for the Harvard linguistic professor George Kingsley Zipf[11,12], is the observation that the frequency of occurrence of some event, (*P*), as a function of the rank (*i*) when the rank is determined by the frequency of occurrence, is a power-law function $P_i \sim 1/i^{\alpha}$ with the exponent α close to unity.

Some characteristics of the distribution can be calculated. Start with the Zipf definition (with α = 1):
P(i) is the probability that page i is requested.

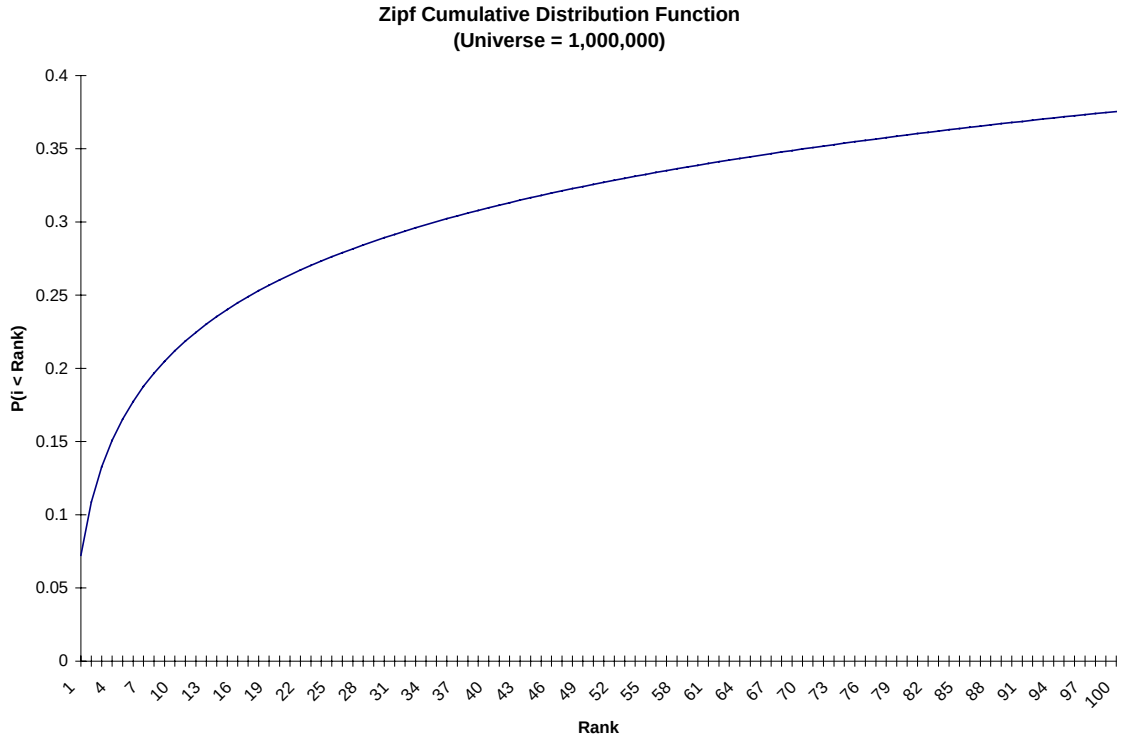
P(1) = X
P(2) = X/2
P(3)=X/3
. . .
P(i) = X/I

To calculate X, observe that the sum of the probabilities is $X \times H_n$ where H_n is the n'th Harmonic sum = 1 + ½ + 1/3 + ¼ + . . . + 1/n. Since the sum of the probabilities must be 1, $X = 1/H_n$. The approximation H_n = natural log of n = ln(n) (especially for large n[13]) is useful here. So p(i) = 1/i ln(n). This is then the probability density function for a universe of n items. The following figure shows this density function for a universe of 1,000,000 items.



The cumulative distribution function can also be calculated. The probability that an item i chosen from a universe of n items has a rank < R is given by:

The sum from 1 to R of P(i) which is approximately ln(R) / ln(n) and is shown in the following chart, for n = 1,000,000



When $\alpha < 1$, the distribution is called Zipf-like, rather than a Zipf distribution. An approximation to the sum of the first n elements of the distribution can be derived as follows:

$$\sum_{i=1}^{n} 1/i^{\alpha} \approx \int_{1}^{n} 1/x^{\alpha}\,dx = \frac{x^{(1-\alpha)}}{(1-\alpha)} = \frac{n^{(1-\alpha)}}{(1-\alpha)} \quad for \alpha < 1$$

So the Cumulative Distribution function for a cache holding k items in an Internet of n objects is

$$k^{(1-\alpha)} / n^{(1-\alpha)} = (k/n)^{(1-\alpha)} \quad \text{for } \alpha < 1$$

References

[1] *Web Proxy Servers*, Ari Luotonen, Prentice Hall, 1997

[2] IPWorX White Paper

[3] Other Web caching General references

[4] P. Barford, A. Bestavros, A. Bradley, and M. E. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications," in World Wide Web, Special Issue on characterization and Performance Evaluation, Vol. 2, pp. 15-28, 1999. Postscript available at: http://cs-www.bu.edu/faculty/crovella/paper-archive/traces98.ps

[5] Pei Cao and Sandy Irani, Cost-aware WWW proxy caching algorithms. In *Proceedings of the 1997 USENIX Symposium on Internet technology and Systems*, pages 193-206, December 1997. Http://www.cs.wisc.edu/~cao/publications.html

[6] HTTP 1.1 RFC 2616

[7] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira, "Characterizing Reference Locality in the WWW", Boston University Computer Science Department, TR-96-11, June 1996. In Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS '96), December 1996.

[8] Lee Breslau, Pei cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: evidence and implications. Technical report, University of Wisconsin-Madison, Department of Computer Science, 1210 West dayton Street, July 1998. Http://www.cs.wisc.edu/~cao/papers/zipf-implications.html

[9] "A Caching Relay for the World Wide Web", Steven Glassman, Systems Research Center, Digital Equipment Corporation, http://www.research.digital.com/SRC/people/Glassman_Steve/cachingTheWeb.html

[10] Zipf Curves and Website Popularity, Sidebar to Jakob Nielsen's column on increasing returns for websites, http://www.useit.com/alertbox/zipf.html

[11] Zipf's Law, Wentian Li, Rockefeller, University, New York, New York, http://linkage.rockefeller.edu/wli/zipf

[12] *The Art of Computer Programming*, Volume 3, Sorting and Searching, Donald E. Knuth, Section 6.1.

[13] *The Art of Computer Programming*, Volume 1, Fundamental Algorithms, Donald E. Knuth, Section 1.2.7