Effect of Speculative Prefetching on Network Load in Distributed Systems

N. J. Tuah
Faculty of Science
Universiti Brunei Darussalam
Gadong BE1410
Brunei Darussalam
jaidi@fos.ubd.edu.bn

M. Kumar
Dept of Computer Science and Engineering
The University of Texas at Arlington
Box 19015 Arlington, TX 76019-0015
kumar@cse.uta.edu

S. Venkatesh
School of Computing
Curtin University of Technology
GPO BOX U1987
WA 6845, Australia
svetha@cs.curtin.edu.au

Abstract

Previous studies in speculative prefetching focus on building and evaluating access models for the purpose of access prediction. This paper, on the other hand, investigates the performance of speculative prefetching. When prefetching is performed speculatively, there is bound to be an increase in the network load. Furthermore, the prefetched items must compete for space with existing cache occupants. These two factors — increased load and eviction of potentially useful cache entries — are considered in the analysis. We obtain the following conclusion: to maximise the improvement in access time, prefetch exclusively all items with access probabilities exceeding a certain threshold.

1. Introduction

Caching and prefetching of data have been used to improve the speed of information access. In caching, copies of remote data are kept locally to reduce access time of repeatedly accessed data [9, 6]. In prefetching, access to remote data is anticipated and the data is fetched before it is required [5]. This is in contrast to *demand fetch* where data is fetched only when it is actually requested.

Prefetching can either be *speculative*, where the knowledge about future accesses is not perfect, or *informed*, where the look-ahead to future accesses is certain. In this paper we investigate speculative prefetching. Previous studies in speculative prefetching (see Section 1.1 Related work) typically focus on building access models and evaluating the performance of such models in predicting future accesses. While these models are important, they do not constitute a complete framework on which to build optimal prefetching strategies. To complement an access model, simple heuristics are usually resorted to, such as to prefetch an item if the probability of its access is larger than a fixed threshold. Though these heuristics might be intuitively sound and their usefulness are empirically confirmed, more analytical treatment is required to understand their performance. This is important considering the fact that prefetching makes use of two valuable assets, namely, bandwidth and memory.

A file fetched from the network requires a certain amount of *retrieval time* measured from the moment of request to the moment of download completion. When the file is demand-fetched, this same amount of time gives the *access time* or the network response time perceived by the application. Prefetching is employed typically to hide the retrieval time so as to give a shorter access time.

From a user's perspective, prefetching is carried out only when his/her network connection is otherwise idle. However, bandwidth is usually shared among several simultaneous users. Prefetching increases network load, reduces the bandwidth pool and, ultimately, affect retrieval times. Thus, indiscriminate use of prefetching may degrade performance.

Indiscriminate prefetching also wastes memory space. Memory allocated for prefetching could have been used for caching. Indeed, caching and prefetching should ideally be integrated so that memory resources can be managed more efficiently. The competition for memory space between prefetched items and existing cache occupants results in what we call the *prefetch-cache interaction*.

In this paper, we study analytically the performance of speculative prefetching, taking into consideration the effect of prefetching on network load. Our analysis also considers two different models for the prefetch-cache interaction. We obtain a condition for prefetching to have the desired effect of reducing the average access time.

1.1. Related work

Access modelling for the purpose of speculative prefetching has received much attention in the literature. Access models are typically constructed based on the assumption that user's requests exhibit repeated patterns. Tait [10] uses file access pattern based on the features of UNIX-style op-

erating system where every program gives rise to a tree of forked processes that access some files. Vitter and Krishnan [13] use data compression techniques to build an access tree that can make optimal predictions if the accesses are generated by a Markov process.

Speculative prefetching has been proposed for improving web access [1, 3, 7]. Padmanabhan and Mogul [7] suggest server-side prediction of document access. The server builds a dependency graph where each link is labelled with the probability of the follow-up access being made. In the ETEL electronic newspaper project [1], the client builds a patterned frequency graph that contains a path for each sequence of accesses. Jiang and Kleinrock [3] combine server-side and client-side prediction for web browsing. They suggest an adaptive prefetching scheme based on a performance model that considers network usage time and user's waiting time.

In [11, 12], performance of speculative prefetching is modelled and analysed, leading to an integrated approach to prefetching and cache replacement decisions. However, the analysis has not considered the effect of prefetching on network load.

Transparent Informed Prefetching (TIP) [8] and Application-Controlled File System (ACFS) [2] integrate prefetching and caching decisions. However, both TIP and ACFS use informed, rather than speculative, prefetching.

1.2. This paper

The rest of the paper is organised as follows. Section 2 describes a multi-user network access model that will be used in our analysis. The section also describes two prefetch-cache interaction models, namely, *models A* and *B*. In Section 3, the average access time for no prefetch is derived. Section 4 derives the average access time when prefetching is performed, and hence formulates the improvement in access time (*access improvement* for short). The access improvement can be maximised by prefetching exclusively all items with individual access probabilities larger than a certain threshold. To obtain this threshold, it is necessary to estimate the cache hit ratio for the case of no prefetch. Section 5 presents a practical method to make this estimate while prefetching may actually be taking place. In Section 6, we briefly talk about *excess retrieval cost*, which is a measure of the increase, due to prefetching, in the amount of retrieval. Section 7 compares models A and B of the prefetch-cache interaction. The conclusions of this paper are presented in Section 8.

Symbols used in this paper are listed in the appendix.

2. Model

This section describes a network access model that will be used in the subsequent analysis. There are two main components in the model, namely, a server access model and a prefetch-cache interaction model. The server access model is similar to the one used by Jiang and Kleinrock [3]. However, they have not considered the prefetch-cache interaction.

Ultimately, we want to obtain the conditions with which prefetching improves the average access time. In particular, we want the *access improvement*, G, to be maximised. Access improvement is defined as

$$G = \bar{t}' - \bar{t}, \tag{1}$$

where $\bar{t}'$ and $\bar{t}$ are the average access times when prefetching is not carried out and when prefetching is carried out, respectively.

2.1. Server access

We consider multiple users accessing the network through a common proxy server. We treat the entire network accessed through the proxy as a server that provides a processor-sharing service for an M/G/1 round-robin queueing system. In this system, the average time to finish a job requiring a service time of x is

$$\bar{r} = x/(1-\rho) \tag{2}$$

where ρ is the system utilisation [4]. In our context, a job is the retrieval of an item. Thus, (2) gives the average retrieval time of an item. Let the average size of an item be $\bar{s}$, and the bandwidth be b. Assume zero network startup latency. The average service time x required to process the retrieval of an item is

$$x = \bar{s}/b \tag{3}$$

The users issue requests at a rate of λ. We assume that this rate is not affected by prefetching (i.e., prefetching is completely transparent to the users). Without prefetching, a proportion h' of these requests results in cache hits and is serviced locally by the clients' caches. The proportion h' is called the cache hit ratio. The complementary value $f' = 1 - h'$ is the cache fault ratio. For cache hits, retrieval times are zero. A cache holds an average of $\bar{n}(\mathcal{C})$ items.

2.2. Prefetch-cache interaction

When prefetching is performed, for each user request an average of $\bar{n}(\mathcal{F})$ items are retrieved in advance. As a result, the hit ratio becomes h. The cache size $\bar{n}(\mathcal{C})$ is assumed large enough to accommodate an arbitrary number of prefetched items. Existing cache occupants may have to be ejected to give room for prefetched items. We shall consider two different ways to model the interaction between caching and prefetching. In *model A*, it is assumed that prefetched items

always eject those that have zero probability of being accessed. This means that prefetching will always increase the cache hit ratio, i.e. $h \geqslant h'$. However, an improvement in cache hit ratio does not imply an improvement in access time. We shall see that, to obtain a positive access improvement, items to be prefetched must meet a certain criterion. In particular, the individual access probabilities of these items must exceed a certain threshold. In *model B*, it is assumed that all the items in the cache contribute uniformly to the value h'. Given the same parameters, the threshold value for the access probabilities is higher in model B than in model A.

2.3. With cache only

Consider the case when prefetch is not performed. Users issue requests at a rate of λ with cache fault ratio f'. The rate of requests to the server is thus $f'\lambda$. Using (2) and (3), the average retrieval time is

$$\bar{r}' = \frac{\bar{s}}{b(1-\rho')} \tag{4}$$

where $\rho' = f'\lambda\bar{s}/b$ is the system utilisation.

The access times for cached items and remote items are 0 and $\bar{r}'$ respectively. Hence, the average access time is

$$\bar{t}' = h'\cdot 0 + (1-h')\bar{r}' = \frac{f'\bar{s}}{b - f'\lambda\bar{s}}. \tag{5}$$

3. Prefetch

We will now derive the access improvement due to prefetching. Consider prefetching, for each user request, an average number of $\bar{n}(\mathcal{F})$ items. For simplicity, assume that all the prefetched files have the same probability p of being accessed. For consistency of probability relations, the number of items with access probability p or larger cannot exceed $\max(n_p)$, where

$$\max(n_p) = \frac{1-h'}{p} = \frac{f'}{p}. \tag{6}$$

As a result of prefetching, the cache hit ratio changes from h' to h. This in turns affects the system utilisation, the retrieval times, and ultimately the access times. The value of h depends not only on which items are prefetched into the cache, but also on which ones are evicted from the cache. It is thus necessary to model the interaction between prefetching and cache replacement. Our analysis is based on the two different models described earlier, namely model A and model B.

3.1. Model A: evict zero-value items

We first consider model A of the prefetch-cache interaction. In this model, it is assumed that we can always find inconsequential items to evict from the cache to give space for incoming ones. As a result, the probability of cache hits is expected to rise from h' to

$$h = h' + \bar{n}(\mathcal{F})p. \tag{7}$$

The rate of users' requests that require demand fetches is $(1-h)\lambda$. The server must service demand fetches as well as prefetches. The effective rate of requests to the server is thus $(1-h)\lambda + \bar{n}(\mathcal{F})\lambda$, giving a utilisation of

$$\rho = (1 - h + \bar{n}(\mathcal{F}))\lambda\bar{s}/b \tag{8}$$

of the server capacity. Using (2) and (3), the average retrieval time is

$$\bar{r} = \frac{\bar{s}}{b - (1 - h + \bar{n}(\mathcal{F}))\lambda\bar{s}}. \tag{9}$$

In the manner similar to the derivation of (5), the average access time is obtained as

$$\bar{t} = (1-h)\bar{r} = \frac{(f' - \bar{n}(\mathcal{F})p)\bar{s}}{b - f'\lambda\bar{s} - \bar{n}(\mathcal{F})(1-p)\lambda\bar{s}}. \tag{10}$$

Substituting (5) and (10) into formula (1) which defines access improvement, we obtain

$$G = \frac{\bar{n}(\mathcal{F})\bar{s}(pb - f'\lambda\bar{s})}{(b - f'\lambda\bar{s})(b - f'\lambda\bar{s} - \bar{n}(\mathcal{F})(1-p)\lambda\bar{s})}. \tag{11}$$

For G in (11) to be positive, the following conditions must be met[1]:

$$\begin{array}{ll} 1. & pb - f'\lambda\bar{s} > 0 \\ 2. & b - f'\lambda\bar{s} > 0 \\ 3. & b - f'\lambda\bar{s} - \bar{n}(\mathcal{F})(1-p)\lambda\bar{s} > 0 \end{array} \tag{12}$$

Condition 2 is met when the system has enough capacity to handle demand fetch requests. Condition 3 is met when the capacity is sufficient to handle prefetch requests as well. Condition 2 is really redundant. We shall see that condition 3 is also redundant. From condition 1, we obtain the threshold value p_{th} for the probability p. To get a positive access improvement, it is necessary that

$$p > p_{\text{th}} = f'\lambda\bar{s}/b = \rho'. \tag{13}$$

Figure 1 shows plots of p_{th} against s for several values of b.

[1] There are other possible ways to make G positive such as to make both the numerator and the denominator negative. However such conditions are only fulfilled when the system load exceeds the system capacity.

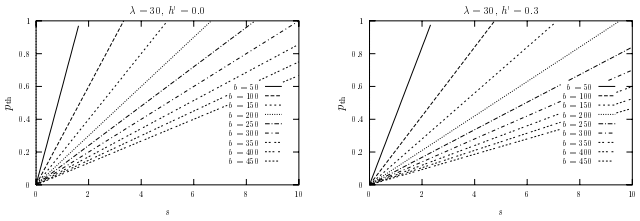

Figure 1. Plots of p_{th} against s for Model A

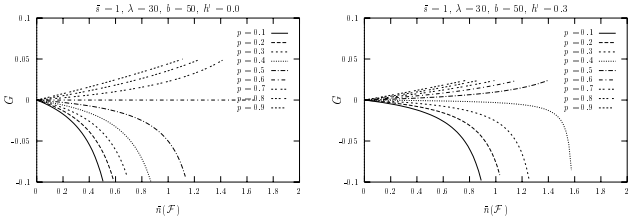

Figure 2. Plots of G against $\bar{n}(\mathcal{F})$ for Model A

From condition 3, we can deduce the limit on the average number of items that should be prefetched for each user request. This is given by $\bar{n}(\mathcal{F}) < \frac{b-f'\lambda\bar{s}}{(1-p)\lambda\bar{s}}$. Consider the least amount of bandwidth that allows useful prefetching of items with access probability p. This amount is just over $f'\lambda\bar{s}/p$ (see condition 1). Under this bandwidth condition, the limit on $\bar{n}(\mathcal{F})$ is

$$\bar{n}(\mathcal{F}) < \frac{f'\lambda\bar{s}/p - f'\lambda\bar{s}}{(1-p)\lambda\bar{s}} = \frac{f'}{p}. \tag{14}$$

Noting (6), and the fact that the constraint in (14) is for the strictest bandwidth condition, we can regard condition 3 as redundant. The implication is that, to make G positive, there is no further restriction on the number of items to prefetch as long as each of the items has an access probability larger than p_{th}.

Figure 2 shows plots of G against $\bar{n}(\mathcal{F})$ for several values of p. Note that each plot is either consistently positive (when $p > p_{\text{th}}$) or consistently negative (when $p < p_{\text{th}}$) or consistently zero (when $p = p_{\text{th}}$).

Evidently, from Figure 2, the positive plots increase monotonously and the negative ones decrease monotonously. It is easy to see analytically that G indeed increases or decreases monotonously for any fixed $p \neq p_{\text{th}}$, as $\bar{n}(\mathcal{F})$ varies from 0 to $\max(n_p)$. Looking at (11), for $p \neq p_{\text{th}}$, if we vary $\bar{n}(\mathcal{F})$ from 0 to $\max(n_p)$, the numerator grows in magnitude but maintains its sign (always positive or always negative), and the denominator shrinks but remains positive. Hence the monotonous change in G. We can therefore conclude the following result:

> *To maximise the access improvement, prefetch exclusively all items with access probability larger than the threshold value p_{th}=ρ', where ρ' is the proportion of the system capacity being utilised when no prefetching is performed.*

3.2. Model B: evict average-value items

We now present for model B the same analysis that we have done for model A. In model B of the prefetch-cache interaction, it is assumed that all the items in the cache, before prefetching, contribute uniformly to the value h'. Since on average there are $\bar{n}(\mathcal{C})$ items in the cache, each is expected to contribute a value of $h'/\bar{n}(\mathcal{C})$ to the cache hit ratio. After prefetching, the cache hit ratio changes to

$$h = h' - \bar{n}(\mathcal{F})h'/\bar{n}(\mathcal{C}) + \bar{n}(\mathcal{F})p \tag{15}$$

Proceeding the way we did for model A, we obtain

$$\rho = (1 - h + \bar{n}(\mathcal{F}))\lambda\bar{s}/b, \tag{16}$$

$$\bar{r} = \frac{\bar{s}}{b - (1 - h + \bar{n}(\mathcal{F}))\lambda\bar{s}}, \tag{17}$$

$$\bar{t} = \frac{(f' + \frac{\bar{n}(\mathcal{F})}{\bar{n}(\mathcal{C})}h' - \bar{n}(\mathcal{F})p)\bar{s}}{b - f'\lambda\bar{s} - \frac{\bar{n}(\mathcal{F})}{\bar{n}(\mathcal{C})}h'\bar{s}\lambda - \bar{n}(\mathcal{F})(1-p)\lambda\bar{s}}, \tag{18}$$

$$G = \frac{\bar{n}(\mathcal{F})\bar{s}(pb - f'\lambda\bar{s} - \frac{bh'}{\bar{n}(\mathcal{C})})}{(b - f'\lambda\bar{s})(b - f'\lambda\bar{s} - \frac{\bar{n}(\mathcal{F})}{\bar{n}(\mathcal{C})}h'\bar{s}\lambda - \bar{n}(\mathcal{F})(1-p)\lambda\bar{s})}. \tag{19}$$

For G in (19) to be positive the following conditions are necessary:

$$\begin{array}{ll} 1. & pb - f'\lambda\bar{s} - bh'/\bar{n}(\mathcal{C}) > 0 \\ 2. & b - f'\lambda\bar{s} > 0 \\ 3. & b - f'\lambda\bar{s} - \frac{\bar{n}(\mathcal{F})}{\bar{n}(\mathcal{C})}h'\bar{s}\lambda - \bar{n}(\mathcal{F})(1-p)\lambda\bar{s} > 0 \end{array} \tag{20}$$

The conditions in (20) are analogous to the previous ones in (12). From condition 1, we can deduce that prefetching will result in a positive access improvement only if p exceeds the threshold value

$$p_{\text{th}} = f'\lambda\bar{s}/b + h'/\bar{n}(\mathcal{C}) = \rho' + h'/\bar{n}(\mathcal{C}). \tag{21}$$

From condition 3, we can deduce the limit on $\bar{n}(\mathcal{F})$. When the bandwidth is just sufficient to make it worthwhile to prefetch items with access probability p, the constraint on $\bar{n}(\mathcal{F})$ is

$$\bar{n}(\mathcal{F}) < f'/(p - h'/\bar{n}(\mathcal{C})). \tag{22}$$

However, $\frac{f'}{p-h'/\bar{n}(\mathcal{C})}$ is larger than $\max(n_p)$ (see (6)). Hence, condition 3 is really redundant. Repeating the same argument that we made for model A, we conclude the following:

> *To maximise the access improvement, prefetch exclusively all items with access probability larger than the threshold value p_{th}=$\rho' + h'/\bar{n}(\mathcal{F})$.*

4. Practical estimation of h'

A prefetch policy that uses the results in the previous section must be able to estimate the value of h'. It would not be sensible to stop prefetching merely to obtain the value of h', which is the cache hit ratio for no prefetch. This value must be obtained even when prefetching is actually taking place. This section describes a method to estimate h'.

Let n_{access} be the total number of user requests. The cache contains *tagged* and *untagged* entries. Let n_{hit} be the total number of requests that are serviced using tagged entries of the cache. The status of cache entries and the numbers n_{access} and n_{hit} are updated using the following algorithm:

- When an item is prefetched:
  1. Insert the item into the cache as untagged
- When a tagged entry is accessed:
  1. $n_{\text{access}} := n_{\text{access}} + 1$
  2. $n_{\text{hit}} := n_{\text{hit}} + 1$
- When an untagged entry is accessed:
  1. $n_{\text{access}} := n_{\text{access}} + 1$
  2. Change the status to tagged
- When a remote item is accessed:
  1. $n_{\text{access}} := n_{\text{access}} + 1$
  2. If the item is admitted to the cache, set its status to tagged.

The value of h' can be estimated as $\frac{n_{\text{hit}}}{n_{\text{access}}}$ if we are using the assumption of model A. For model B, the estimate is $\frac{n_{\text{hit}}}{n_{\text{access}}} \times \frac{\bar{n}(\mathcal{C})}{\bar{n}(\mathcal{C})-\bar{n}(\mathcal{F})}$.

5. Excess retrieval cost

When prefetching is carried out based on speculation, there is bound to be an increase in the amount of retrieval due to incorrect prefetches. Let the *excess retrieval cost*, C, be defined as

$$C = R - R', \tag{23}$$

where R and R' are the retrieval times for prefetch and no prefetch, respectively. In this brief section, we will derive a formula for C that takes into account the effect of increased network load on retrieval times. We will keep the formula generic in regard to prefetch-cache interaction.

The system utilisation can be expressed as

$$\rho = \bar{n}(\mathcal{R})\lambda\bar{s}/b \tag{24}$$

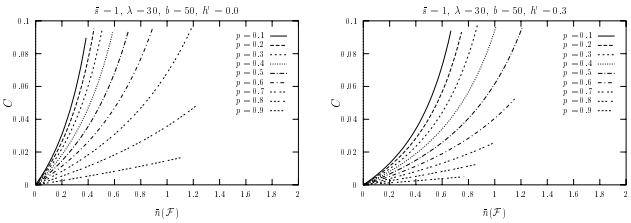

Figure 3. Plots of C against $\bar{n}(\mathcal{F})$ for Model A

where $\bar{n}(\mathcal{R})$ is the number of retrieved items per user request. Using (2) and (24), the retrieval time per user request is

$$R = \bar{n}(\mathcal{R})\bar{r} = \rho/(\lambda(1-\rho)). \tag{25}$$

Equation (25) is general. However, just to be explicit the retrieval time per user request when no prefetch is performed can be stated as

$$R' = \bar{n}'(\mathcal{R})\bar{r}' = \rho'/(\lambda(1-\rho')). \tag{26}$$

Thus,

$$C = R - R' = \frac{\rho - \rho'}{\lambda(1-\rho)(1-\rho')}. \tag{27}$$

From (27), we note that prefetching is subjected to *load impedance*. By this, we mean that prefetching an item when the system load is high costs more than prefetching the same item during low system load. Figure 3 shows plots of C as a function of $\bar{n}(\mathcal{F})$ for the same set of parameters used to plot Figure 2.

6. The two models compared

Comparing the results for the two prefetch-cache interaction models, we note the following:

- In both models, once the threshold condition for the access probability is met, there is virtually no further restriction on the number of items to prefetch.
- Comparing (13) and (21), and noting that $h' \leqslant 1$, the difference in the values of the threshold p_{th} between the two models is at most $\frac{1}{\bar{n}(\mathcal{C})}$. To make this difference significant, either the bandwidth capacity is high compared to the demand (small ρ'), or the cache is comparatively meagre (small $\bar{n}(\mathcal{C})$). Prefetching is less necessary when ρ' is small.
- The values of h (and hence ρ, $\bar{r}$, $\bar{t}$, G and C) in both models are approximately equal when $\bar{n}(\mathcal{C}) \gg \bar{n}(\mathcal{F})$.

From the above observations, we can see that model A can approximate model B in the sense that prefetch decisions

based on the former are likely to be very similar to decisions based on the latter.

Let us ponder on a more realistic model, *model AB*. In this model, every item in the cache has a (possibly zero) contribution to the value of h'. In practice, this contribution is unlikely to be uniform among the cached items. Thus, inevitably we can always find an item to evict whose access probability is less than $h'/\bar{n}(\mathcal{C})$. If we continue the analysis, we will obtain results that are between those for models A and B. Obviously, if model A can approximate model B, the former can approximate model AB even better. We can thus conclude that model A, despite its simplistic assumption, is a reasonable approximation of the more realistic model. A particular advantage of model A is that it has one less parameter. Specifically, $\bar{n}(\mathcal{C})$ is assumed to be large but its value does not have to be known.

7. Conclusions

In this paper, we have analysed the performance of speculative prefetching, taking into account the effect of prefetching on network load. We also have considered two different models for prefetch-cache interaction. In both models, we have arrived at the same conclusion: to maximise the improvement in access time, prefetch exclusively all items with access probabilities exceeding a certain threshold. To calculate the threshold, the value of the cache hit ratio for no prefetch, h', must be obtained. However, the value of h' would be hypothetical if prefetching is actually taking place! We have presented a possible method to estimate h'. We have also briefly discussed excess retrieval cost, which is defined as the additional network time utilised for prefetching items. Our studies show that prefetching the same item will have different costs under different network loads. We are investigating the application of this work in addressing QoS issues of multimedia access in wired as well as wireless networks. Furthermore, these results can also be applied to prefetching files or data in distributed computing applications.

Appendix: List of Symbols

In this paper, symbols marked with a prime are for the case of no prefetch. For example, h' is the cache hit ratio when no prefetch is carried out.

b	: bandwidth
R, R'	: retrieval time per user request
$\bar{r}, \bar{r}'$	: average retrieval time of an item
$\bar{s}$	: average item size
$\bar{t}, \bar{t}'$	: average access time
h, h'	: cache hit ratio
f'	: cache fault ratio
ρ, ρ'	: server utilization
$\bar{n}(\mathcal{F})$	: number of prefetched items per user request
$\bar{n}(\mathcal{C})$	: average number of items in a user's cache
$\bar{n}(\mathcal{R}), \bar{n}'(\mathcal{R})$	: number of retrievals per user request
λ	: rate of user requests
p	: access probability
p_{th}	: threshold value for access probability
$\max(n_p)$	: maximum number of items with access probabilities p or larger

References

[1] M. Banâtre, V. Issarny, F. Leleu, and B. Charpiot. Providing quality of service over the web: A newspaper-based approach. *Computer Networks*, 29(08–13):1457–1465, Sept. 1997.

[2] P. Cao. *Application-Controlled File Caching and Prefetching*. PhD thesis, Department of Computer Science, Princeton University, Jan. 1996.

[3] Z. Jiang and L. Kleinrock. An adaptive network prefetch scheme. *IEEE Journal on Selected Areas in Communications*, 16(3):358–368, Apr. 1998.

[4] L. Kleinrock. *Queueing Systems Vol 2: Computer Applications*. Wiley, 1975.

[5] H. Lei and D. Duchamp. An analytical approach to file prefetching. In *Proc USENIX Annual Technical Conf*, Jan. 1997.

[6] M. N. Nelson, B. B. Welch, and J. K. Ousterhout. Caching in the Sprite network file system. *ACM Trans on Computer Systems*, 6(1), 1988.

[7] V. N. Padmanabhan and J. C. Mogul. Using predictive prefetching to improve World Wide Web latency. *ACM SIGCOMM Computer Communication Review*, pages 22–36, July 1996.

[8] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka. Informed prefetching and caching. In *Proc 15th ACM Symposium on Operating System Principles*, pages 79–95, Dec. 1995.

[9] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. Design and implementation of the Sun network file system. In *Proc Summer 1985 USENIX Conf*, pages 119–130, June 1985.

[10] C. D. Tait. *A File System for Mobile Computing*. PhD thesis, Graduate School of Arts and Sciences, Columbia University, 1993.

[11] N. J. Tuah, M. Kumar, and S. Venkatesh. Investigation of a prefetch model for low bandwidth networks. In S. K. Das, editor, *1st ACM Int Workshop on Wireless Mobile Multimedia*, pages 38–47, Oct. 1998.

[12] N. J. Tuah, M. Kumar, and S. Venkatesh. Performance model of speculative prefetching in distributed information systems. In *Proceedings IPPS/SPDP 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, pages 75–80, Apr. 1999.

[13] J. S. Vitter and P. Krishnan. Optimal prefetching via data compression. In *IEEE 32nd Annual Symposiums on Foundation of Computer Science*, pages 121–130, 1991.