预取技术研究进展*)

曹新平[1] 刘美华[1] 韩 真[2] 古志民[1] 张建鑫[1]
(北京理工大学计算机科学工程系 北京100081)[1] (中国农业银行海淀支行 北京100081)[2]

State of the Art of Prefechting Technique

CAO Xin-Ping LIU Mei-Hua HAN Zhen GU Zhi-Min ZHANG Jian-Xin
(Dep. of Computer Science and Engeering, Beijing Institute of Technology, Beijing 100081)
(Beijing Haidian Subbranch of Agriculture bank of China, Beijing 100081)
E-mail: caoxinpinghz@163.com

Abstract World Wide Web(WWW) services have grown to levels where significant delays are expected to happen. Technology like prefetching are likely to help users to personalize their needs, reducing their waiting times. This paper firstly describes the architecture of prefetching, then classifies them into three types: based on branch model, based on tree model and others and presents profoundly the basic ideas of some existing prefetching algorithms. Next, several models for controlling the prefetching are introduced. At last, the trend and course concerning the prefetching algorithms are concluded.

Keywords Prefetching, Web cache, Access latency, QoS, Algorithm

1 引言

随着 Internet 的飞速发展，Internet 的用户数也呈指数级增长，使得网络负载日趋严重，加之网络带宽的不足，导致用户在浏览网站信息时无法忍受过长的延迟时间。因此，减少网络时延已经成为改善 Internet 应用环境的主要目标。

解决网络延迟的一种行之有效的方法是采用网络缓存技术。但据最近的研究表明，Web 缓存代理服务器的命中率不超过50%[1,2,23~25]。为了突破这个上限，可以采用预取技术。预取的含义是在用户需要一些页面之前就把这些页面从源服务器取来并保存在本地缓存中，以供用户将来使用。预取机制使得代理服务器的命中率达到60%～80%[3,4]。一般来说，预取包括三个部分：预取系统结构，预取算法和预取控制。

2 预取系统结构

预取存在三种情况：客户和服务器之间、客户和代理之间以及代理和服务器之间[28]。Padamanabhan 和 Mogul 研究了客户和服务器系统的预取策略[18]，预取系统结构如图1所示。在服务器端，存在两种类型的用户进程：HTTP 后台进程和预测引擎进程。HTTP 后台进程负责处理客户请求，实现 WWW 服务。预测引擎则负责产生预测信息。在客户端，预取引擎负责根据服务器发来的预测信息决定预取哪些网页。客户和代理之间、代理和服务器之间的预取系统结构与上图基本类似，这里不作说明。

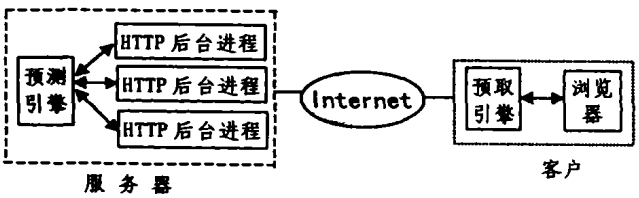

图1 预取系统结构图

3 预取算法的基本思想

目前研究的预取算法主要有以下三种：

3.1 基于分支模型的预取算法

分支模型的意思是根据当前访问节点预取下一个节点，而下一个节点可能在一个节点集中。模型结构如图2所示。文[4]把用户的请求和使用分离开来，通过对客户当前正在访问的或刚才访问过的页面的 HTML 进行语法分析，把这些页面的链接节点收集起来，然后把这些链接节点预取过来，存放在代理里以满足用户的请求。而文[31]通过分析 HTML 页面，把内嵌元素特别是很大的图像文件预取过来。

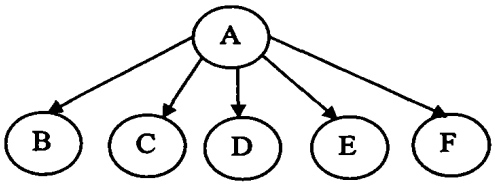

图2 分支模型

该算法通过分析页面的链接节点来预取页面，容易实现。但是，如果用户当前打开了很多页面，通过语义分析后会产生很多链接节点，根据这些链接节点进行预取时，可能会导致用户服务质量很差。故文[13,14]加入了一些历史信息来帮助预取那些链接节点。

文[9,10,15,17,18]定义用户访问节点 A 后访问节点 B 的概率为：

$$p(B/A)=C_{(A,B)}/C_B \quad (1)$$

其中 C_A 表示资源计数器和 $C_{(A,B)}$ 表示链接计数器。由于用户

*)出国留学基金资助。曹新平 研究生，研究方向：预取技术。古志民 教授。

又. Web 的访问具有一定的规律，且具有历史性和相对集中的爱好，因此文[5,6]提出了基于组的兴趣和访问行为对将要访问的节点进行预测。该算法定义的访问概率如下：

$$P=\beta P_u+(1-\beta)P_g \tag{2}$$

其中 $\beta(0<\beta<1)$为个人访问概率的权值，组的成员的兴趣差别越大，β 取值越大，P_u 为个人访问概率，P_g 为组访问概率。根据节点的 P 值来决定是否对该节点进行预取。

对该算法的链接计数器 $C_{(A,B)}$的统计是比较困难的，因为用户的访问路径不同会使得 $C_{(A,B)}$很小。当用户的 Web 访问历史资料很少时，无论怎样调节 β 值，都不可能真正反应用户的访问兴趣。另外，基于历史页面链接的算法需要很大的空间来存放有关链接的数据结构，这对缓存器的空间是个威胁。

在访问概率的基础上，还得考虑预取该节点的时间。如果预取该节点的时间很长，而且以后没有被访问，那么就会严重影响网络的性能。因此，我们把访问概率和预取该节点的时间结合起来作为该节点的预取优先数。下面给出定理来计算节点的优先数[33]。

定理1(分支决策)　假设 $A=n_c$ 是当前访问的节点，在访问该节点后即将要访问的是候选节点集 $n_1,n_2,n_3,\cdots,n_n$ 中的一个节点，T_i 为下载 n_i 的估计时间，$P[a_{n+1}=n_i|a_n=A]$是条件链接转移概率，平均延迟最小的充分条件为按照优先数 Q_i 的高低预取链接节点，这里

$$Q_i=P[a_{n+1}=n_i|a_n=A]/T_i \tag{3}$$

3.2　基于树模型预取算法

树模型就是把以前访问的节点按照访问的顺序组成一棵树，结构如图3所示。特别地，我们把一个链也当成一棵树。文[11,12]从服务器挖掘用户的访问日志得到用户的访问序列(称之为 Markov 链)，把用户当前的访问序列同挖掘得到的序列匹配，然后把这些已匹配的序列的下一个节点都预取过来。文[8,16]利用数据挖掘技术挖掘用户的兴趣关联规则，作为对用户即将访问的节点进行预取的依据。文[20]根据服务器日志首先构造访问树，然后把它们存起来。在客户访问时，通过计算当前访问树与存起来的访问树的兼容性，如果兼容性都比较高，则取最近存的那棵访问树，然后把该树没有被访问的节点预取过来。文[21]从服务器的日志得到用户的访问序列，然后把这些访问序列构造成权值树，把当前的用户的访问序列通过逐步后退编码算法与权值树里的访问序列匹配，选择匹配上的最长的序列作为预取序列，预取该序列中的一个节点，该节点是用户当前访问的节点，它在该预取序列的下一个节点。文[22]的预取算法包含三个参数：M(用户已访问过的节点)，L(该算法预取的层次数)和 T(阈值，只预取超过该值的节点)。该算法维持一棵访问树，该树的节点访问概率都大于 T，树的深度为 M+L+1，第一层为根节点。通过用户

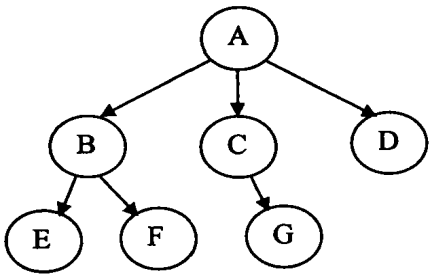

图3　树模型

当前访问的序列(只有 M 个页面)匹配该树中的子树的第二层到第 M 层，然后把第 M 层以下的节点预取过来。

同样，我们可计算树模型里的节点的优先数，下面给出该定理：

定理2(树决策)　设 $A=n_0$为当前访问的节点，$\{n_1,n_2,n_3,\cdots,n_d\}$是构成树的一条路径的节点序列，节点 n_d 的层次数为 d，T_d 为该节点的估计下载时间值，那么该节点的优先数 Q_d 为：

$$Q_d=\prod_{i=c}^{d}\Pr[a_{i+1}=n_{i+1}|a_i=n_i] \tag{4}$$

3.3　其它的预取算法

文[3]使用代理服务器定期地统计它上面的网页的访问次数，并选取访问次数较多的网页组成流行页面集。然后根据客户最近发出的请求量的大小，从每个服务器上的流行页面集中预取相当于用户最近发出的请求量的页面放在缓存或直接送给用户。当客户的请求量很大时或者缓存里的页面的重复利用率很高时，预取得到的效果比较好。

基于流行度的预取算法简单，容易实现。该算法通过二级代理聚集用户的请求，使得网络的通信量的增加不超过20%，并且只预取那些流行的页面，降低了预取无用页面的概率。然而服务器的页面远远不止这些流行度较大的页面，对于那些流行度较小的网页来说是不公平的，而这些页面可能是某些用户当前感兴趣的东西。另外，它把网络上的服务器的负载转移到代理上，这可能导致代理的超负荷运作，需要解决代理和服务器之间负荷的平衡问题。对那种机械适应方式，即用户最近发出了多少个请求，就从每个服务器上去预取相应个数的网页，这不但会使网络通信量急剧增大，而且如果用户的请求量突然增大时，还会使得用户的命中率降低。更为重要的是：它不是交互式的，不能灵活地根据用户当前的请求预测以后的请求，这样用户的连续访问将得不到较好的响应。

节点的生存期是节点两次相继被修改之间的时间。文[27]设节点 i 的生存期为 l_i，该节点已被访问概率为 p_i，用户的请求到达率为 a，则该节点即将访问的概率为：

$$P=1-(1-p_i)^{a\times l_i} \tag{5}$$

上式中，$a\times l_i$ 为在节点 i 的生存期内请求到达数，$(1-p_i)^{a\times l_i}$为在节点 i 的生存期内没有请求节点 i 的概率。

文[34]定义节点 i 请求数如下：

$$P=a\times p_i\times l_i \tag{6}$$

上式中的字母表示的意义同(5)式。利用 P 值来预取节点。

还有一些文献通过硬件来实现预取机制如文[35]。

上面所说的算法除文[4]外都是通过统计历史信息来进行预测的，这种用历史信息来做预测的算法有4个缺点：1)通过统计来预测用户即将发出的请求是很困难的，而且用户即将访问的页面只占预取页面的一小部分；2)用户的兴趣通常变得很快，这使得算法的效率较低；3)要想得到较高的命中率，需要通过很长的时间来进行观察和统计，这导致立即把握用户的新趋向很困难。4)存储历史信息和计算概率的花销很大。另外，以上算法没有考虑到用户的个人兴趣问题，尽管提高了被经常访问的页面的命中率，但没有全面减低网络的延迟，而且在一定程度上增加了通讯量[19]，因而预取的效果不尽人意。

4　预取的控制

预取的另一方面是对预取的控制[9]，必须限制预取使用的网络资源和进行预取的时间，避免对其它网络应用的不良

影响。

4.1 门限算法

该算法的核心思想是评价预取节点的尺度是否超过某值。文[9,10,15,5,6]定义预取的代价由延迟代价 α_T 和系统资源函数代价 α_B 组成。其中，文[9]分两个系统：单用户系统和多用户系统。而文[10,15,5,6]只包含多用户系统。

4.1.1 单用户系统 该系统只包含一个用户和多个服务器。如果即将访问的节点不在当地的缓存里，那么要预取该节点的访问概率必须满足以下条件：

$$p_i > 1/\frac{\alpha_T}{\alpha_B b_i}(1+\frac{s_{0_i}}{s_i})+1 \quad (7)$$

上式中，p_i 为节点 i 的访问概率，b_i 为到达第 i 个服务器的带宽，s_i 为第 i 个节点的大小，$s_{0_i}=b_i \cdot t_{0_i}$，$t_{0_i}$ 为检索第 i 个节点的启动时间。

4.1.2 多用户系统 该系统由多个用户和一个服务器组成。在系统启动时间为零的情况下，预取的节点的访问概率必须满足以下条件：

$$p \geqslant 1-(1-\rho)\frac{\alpha_T}{\alpha_B}/(1-\rho)^2 b+\frac{\alpha_T}{\alpha_B} \quad (8)$$

在系统启动时间不为零和 α_T 相同的情况下，预取节点的访问概率满足的条件为：

$$p \geqslant 1-(1-\rho)(1+\frac{s_0}{s})\frac{\alpha_T}{\alpha_B}/((1-\rho)^2 b+(1+\frac{s_0}{s})\frac{\alpha_T}{\alpha_B}) \quad (9)$$

在 α_T 不同的情况下，预取节点的访问概率满足的条件为：

$$p \geqslant 1-\frac{(1-\rho)\frac{\alpha_T}{\alpha_B}}{(1-\rho)^2 b+\frac{\alpha_{Tk}}{\alpha_B}(1-\rho)+\frac{s}{b\alpha_B}\sum_{i=1}^{n}\lambda_{ii}\alpha_{T_i}} \quad (10)$$

该式是在系统启动时间为零的情况下得到的。其中 ρ 为系统负载，s 为文件平均大小，b 为系统容量，α_{T_k} 是第 k 类用户的延迟代价，λ_{ii} 为 i 类用户的正常到达率。

4.2 对预取的传输速率进行控制

文[19]对预取的传输速率进行控制的实现是基于这种事实：在预取一个对象时，不必以网络能维持的最大速率传送该对象，而只要在用户使用前以足够的速率把该对象传输完毕即可。因此，对预取传输速率控制的关键是在预取对象时，把传输率降低到这样一个水平：预取应尽可能地早，而该预取对象的最后一个字节在用户使用前传输完毕即可。我们限制了TCP使用的最大窗口尺寸来确定预取时的传输速率。窗口尺寸的大小为：

$$W=[P \cdot R/T] \quad (11)$$

上式中，p 为传输该文档的分组数，R 为服务器和客户机之间的往返时间，T 为前一个文档请求文档结束和下一个文档请求开始之间的时间间隙。另外，该文献还介绍了一种减少分组的有效负载来控制预取的速率。

结论 在分布式系统中如Web，预取是一种很好的减少时延的技术，当前有很多实现某种预取算法的商业系统如：CacheFlow预取内嵌的资源；NetSonic，PeakJet 2000和Webcelerator预取当前页的链接节点，也已有研究把预取算法用于无线网通信中[32,10]。

随着预取算法的进一步研究，预取算法将全面降低网络时延，极大减少网络通讯量，越来越重视个人的兴趣，使历史信息只在一定范围内起作用，而不像今天这样占主导地位。

参考文献

1 Abrams M, et al. Caching Proxies: Limitations and Potential. http://ei.cs.vt.edu/succeed/WWW4.html

2 Chankhunted A, et al. A Hierarchical Internet Object Cache. UNIX 1996 TECHNICAL CONFERENCRE, http://Excalibur.usc.edu/cache-hrml/cache.html

3 Marcatos E P, Chronaki C E. A Top-10 Approach to Prefetching the Web. In: Proc. of the Eighth Annual Conf. of the Internet Society(INET'98), Geneva. Switzerland. July 1998

4 Chinen K-i, Yamaguchi S. An Interactive Prefetching Proxy Server for Improvement of WWW Latency. In: Proc. of the Seventh Annual Conf. of the Internet Society(INET'97). Kuala Lumpur. June 1997

5 金志刚，张钢，舒炎泰. 基于网络性能的智能Web加速技术——缓存与预取. 计算机研究与发展，2001(8)

6 赵政，张钢，杨洁，王松，舒炎泰. Web智能代理的预取技术和缓存技术. 天津大学学报，2001(9)

7 谭琼，李晓黎，史忠植. 一种实现搜索引擎个性化的方法. 计算机科学. 2002,29(1)

8 Zhang Wei-Feng, Xu Bao-Wen, Chu William C, Yang Hong-Ji. Application of data mining in Web pre-fetching. In: Proc IEEE MSE2000, 2000

9 Jiang Z, Kleinrock L. An adaptive network prefetch scheme. IEEE Journal on Selected Areas in Communications, 1998, 16(3): 358～368

10 Jiang Z, Kleinrock L. Web prefetching in a mobile environment. IEEE Int. Conference on Communications, 1998, 5(5): 25～34

11 Su Z, Yang Q, Lu Y, Zhang H. Whatnext: A prediction system for web requests using n-gram sequence models. In: Proc. of the First Intl. Conf. on Web Information System and Engineering Conference, Hong Kong, June 2000. 200～207

12 Yang Q, Zhang H H, Li Tianyi. Mining Web Logs for Prediction Models in WWW Caching and Prefetching. In: The Seventh ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining KDD'01, Aug. San Francisco, California, USA, 2001

13 Duchamp D. Prefetching hyperlinks. In: Prec. of the Second USENIX Symposium on Internet Technologies and Systems (USITS'99), Boulder, CO, Oct. 1999

14 Davison B D. Predicting web action from html content. HT'02, College Park, Maryland, USA. 2002

15 Zhimei J, Kleinrock L. Prefetching links on the WWW. In: Proc. of the 1997 IEEE Intl. Conf. on Communications. Towards the Knowledge Millennium. (ICC '97). Montreal, Que., Canada. 1997. 483～489

16 徐保文，张卫丰. 数据挖掘技术在web预取中的应用. 计算机学报，2001，24(4)

17 Conhen E, Kristhnamurthy B, Rexford J. Efficient Algorithms for Predicting Requests to Web Servers. In IEEE INFOCOM, Anchorage, AK, Apr. 2001

18 Padmanabhan V N, Mogul J C. Using predictive prefetching to improve World Wide Web latency. Comput. Commun. Rev., July 1996. 22～36

19 Crovella M, Barford P. The network effects of prefetching. In: Proc. IEEE INFOCOM, April 1998

20 Lei H, Duchamp D. An analytical approach to file prefetching. In: Proc. of USENIX 1997 Annual echnical Conf. USENIX, Jan. 1997

21 Schechter S, Krishnan M, Smith M D. Using path profiles to predict http requests. In: Proc. of the Seventh Intl. World Wide Web Conf. Brisbane, Australia

（下转第55页）

的处理详见文[15]。

3 安全多方计算协议的研究方向

概括起来，目前众多的研究者的研究主要集中在如下一些方面：(1)一般化的安全多方计算协议：这类研究的目的是设计一种高效的、安全的、能够计算任意函数的安全多方计算协议，希望能够通过这样一种协议一劳永逸地解决所有的涉及到安全多方计算的问题。(2)对一般化的安全多方计算协议进行剪裁：这类研究注意到如果一个协议是广泛适用的，那么必然会牺牲一些性能上的代价来满足其广泛使用性。针对于某一个具体的应用，如电子选举，对一般化的安全多方计算协议进行一点剪裁，去掉一些对某个具体应用没有意义的部分，可以大大地提高效率。这类研究主要研究的是如何将安全多方计算的理论和技术应用到一个具体的应用中去。(3)安全多方计算的定义：安全多方计算的目的和应该具备的性质都为在这一领域研究的学者熟知了，但是安全多方计算至今还缺乏一个为大家所认同的、完备的定义。这主要是因为安全多方计算协议的构造形式可以有多种，各个学者研究的安全模型也是不一样的。(4)新的安全多方计算协议的构造方法：目前大部分学者提出的安全多方计算协议都使用了 VSS 子协议作为其构造基石，因而这类协议在大体结构上都是类似的。有没有其他的方法来构造安全多方计算协议?这也是一些学者的研究内容。(5)安全多方计算协议生成器：许多安全多方计算协议的研究在研究了如何安全地计算域上定义的基本运算就停止了，因为这时从理论上讲，任何函数都可以被安全地计算。但是这离实际应用还有一步需要跨越，多方计算协议生成器的作用就是弥补这最后的一步。安全多方计算协议生成器的输入是任意函数和如何计算域上定义的基本运算的子协议，输出是安全计算该函数的协议。许多学者提到了对方计算协议生成器，但是没有对之进行细致的研究。

参考文献

1 Yao A C. Protocols for secure computations. In:Proc. 23rd IEEE Symp. On the Foundation of Computer Science, IEEE, 1982. 160～164

2 Goldreich O, Micali S, Wigderson A. How to play any mental game. In:Proc. 19th ACM Symp. On the Theory of Computing, 1987. 218～229

3 Chaum D, Crepeau C, Damgard I. Multiparty unconditionally secure protocols (extended abstract). In:Proc. 20th ACM Symp. On the Theory of Computing, 1988. 11～19

4 Goldwasser S, Levin L. Fair computation of general functions in presence of immoral majority. In Advances in Cryptology-CRYPTO'90 volume 537 of LNCS. Springer-Verlag, 1990

5 Goldreich O, Goldwasser S, Linial N. Fault-Tolerant Computation in the Full Information Model. 32nd FOCS, 1991. 447～457

6 Ostrovsky R, Yung M. How to withstand mobile virus attacks. In:Proc. of the 10th Annual ACM Symposium on Principles of Distributed Computing, 1991. 51～59

7 Micali S, Rogaway P. Secure Computation. CRYPTO, 1991

8 Beaver D. Foundations of Secure Interactive Computing. CRYPTO, 1991

9 Goldreich O. Secure multi-party computation(working draft, Version 1.1). 1998

10 Chor B, Goldwasser S, Micali S, Awerbuch B. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In: Proc. 26th Annual Symposium on the Foundations of Computer Science, IEEE, 1985. 383～395

11 Gennaro R, Rabin M, Rabin T. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In:Proc. of the 1998 ACM Symposium on Principles of Distributed Computing

12 Matthew, Franklin, Habert S. Joint encryption and message-efficient secure computation. Journal of Cryptology, 1996, 9(4):217～232

13 Paillier P. Public-key cryptosystems based on composite degree residue classes. In:Michael Wiener, ed. Advances in Cryptology-EuroCrypt'99, Berlin, 1999. 223～238

14 Cramer R, Damgard I, Nielsen J B. Multiparty Computation form Threshold Homomorphic Encryption. BRICS. June, 2000

15 Jakobsson M, Juels A. Mix and Match:Secure Function Evaluation via Ciphertexts

(上接第30页)

22 Fan Li, Cao Pei, Jacobson Q. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. In:Proc. of the ACM SIGMETRICS'99, Atlanta, Georgia, May 1999

23 Douglis F, Feldmann A, Krishnamurthy B, Mogul J. Rate of change and other metrics: a live study of the world wide web. In: Proc. of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, Dec. 1997

24 Feldmann A, C'aceres R, Douglis F, Glass G, Rabinovich M. Performance of Web proxy caching in heterogeneous bandwidth environments. In:Proc. of the IEEE INFOCOM'99 Conf. 1999

25 Kroeger T M, Long D D E, Mogul J C. Exploring the bounds of web latency reduction from caching and prefetching. In:Proc. of the USENIX Symposium on Internet Technologies and Systems, Dec. 1997. 13～22

26 Thompson K, Miller G, Wilder R. Wide-area Internet traffic patterns and characteristics. In: Proc. 3td Int'l. conf. Web Caching, 1998

27 Venkataramani A, et al. The potential costs and benefits of long-term prefetching for contentdistribution. In: Sixth International Workshop on Web Caching and Content Distribution, June 2001

28 Cao P, Zhang J, Beach K. Active cache: caching dynamic contents on the web. In: Proc. IFIP Int'l. Conf. Dist. Sys. Platforms and open Dist. processing, 1998

29 Gwertzman J S. Autonomous replication across wide-area internetworks. Thesis, Harvaed College, Cambridge, MA, 1995

30 Davison B D. Assertion: prefetching with get is not good. In: the Proc. of the 6th Intl. Workshop on Web Caching and Content Distrribution, June. 2001. 20～22

31 Dodge R, Menasce D A. Prefetching inlines to improve web server latency. In: the Proc. of the 1998 Computer Measurement Group Conf. Anaheim, CA, Dec. 1998. 6～11

32 Gitzenis S, Bambos N. Power-controlled data prefetching/caching in wireless packet network. IEEE. 2002

33 Khan J I, Tao Qingping. Partial Prefetch for Faster Surfing in Composite Hypermedia. In: the Proc. of the 3rd USENIX Symposium on Internet Technologies USITS'01San Francisco March 2001. 13～24

34 Jiang Yingyin, Wu Min-You, Shu Wei. Web Prefetching: Costs, Benefits and Performance

35 An-Chow Lai, Cem Fide and Babak Falsafi. In:Proc. Of the 28th Annual International Symposium on Computer Architecture 2002