マルチメディア通信と分散処理 74－33
グループウェア 15－33
（1996. 1. 26）

WWWプロキシサーバのログに基づいたキャッシュ置き換えアルゴリズムの評価

大澤 範高, 早野 文孝, 弓場 敏嗣, 箱崎 勝也
電気通信大学 情報システム学研究科

World Wide Web (WWW) のアクセスパタンに適したプロキシサーバキャッシュ置き換えアルゴリズムとして世代型の置き換えを提案し、その効果を評価した。最長未使用 (LRU) アルゴリズムと比較して、エントリあたりで4ポイント程度のヒット率向上が得られることがわかった。これは、先入れ先出し (FIFO) と比較した際の LRU による向上と同程度以上である。また、データに関するキャッシングに加えて、WWW アクセスのコネクションのキャッシングについても評価し、コネクションのキャッシングの効果が高いことを示す。これらの評価は、プロキシサーバのログを基に行なった。

Evaluation of Replacement Policies in WWW Cache Based on Logs from Proxy Server

Noritaka OSAWA, Fumitaka HAYANO, Toshitsugu YUBA, Katsuya HAKOZAKI
Graduate School of Information Systems
The University of Electro-Communications

We propose and evaluate the generational replacement algorithm that is suitable for access patterns to the World Wide Web (WWW) proxy server cache. Using the algorithm, the hit rate of entries is improved by about 4 point from the Least Recently Used (LRU) algorithm. This improvement is roughly equal to that of LRU to the First-In First-Out (FIFO) algorithm. In addition to data caching, the effect of caching of connections to access the WWW pages is evaluated. We show that caching of connections is effective. These evaluation is based on logs from a proxy server.

1 はじめに

Internet では World Wide Web (WWW) の利用が急速に広がっており、ネットワークトラフィックの大きな部分を WWW アクセスが占めるようになっている。また、ユーザの WWW アクセスの待ち時間や、WWW サーバの負荷が問題となっている。

これらの問題を解決するためには、キャッシングが有効であると考えられる。良くアクセスされるデータをクライアントに近い側でキャッシングすることによって、ネットワークトラフィックの削減やアクセスレイテンシ (遅延) の短縮を図ることができる。また、サーバへのアクセスも減少するので、WWW サーバの負荷も軽減される。

WWW では、防火壁 (firewall) を越えたアクセスを行なえるようにプロキシサーバが導入され、プロキシがキャッシング機能を持つようになっている [6][8]。プロキシにおけるキャッシングに関する研究は、[1] [2] [4] [10] 等でも行なわれているが、WWW のアクセスの拡大に伴った大量のデータに基づいた解析は十分とは言えない。

本研究では、電気通信大学の総合情報処理センターで運用されている WWW プロキシサーバのログを基にして、キャッシュの置き換えアルゴリズムによって WWW のネットワークトラフィック削減やアクセスレイテンシ短縮の効果の指標となるキャッシュのヒット率にどのような違いがあるかを評価する。

2 プロキシサーバのログ

キャッシュ置き換えアルゴリズムの評価に入る前に、電気通信大学の総合情報処理センタープロキシサーバとそのログのデータについて説明する。

2.1 サーバと利用者

電気通信大学の総合情報処理センターでは、CERN httpd をプロキシサーバとして運用している。講義、演習等で利用される総合情報処理センターの教育系ワークステーションは学外とは直接通信できないので、学外の WWW ページを参照するためには、プロキシサーバを利用しなければならない。このため、教育系ワークステーションから WWW を利用している学生は、全員がプロキシサーバを利用していると考えられる。この結果、学内のページをアクセスする場合もプロキシサーバが利用されている。また、電気通信大学の総合情報処理センターのプロキシサーバは、独自のプロキシサーバを運用していない学科/専攻や研究室からも利用されている。

総合情報処理センター教育系ワークステーションでは mosaic が WWW クライアントとして利用されている。mosaic は、セッションを超えたクライアントでのキャッシングは行なわない。教育系以外の学内の計算機では、mosaic の他に netscape などが利用されている。netscape は、セッションを超えたキャッシングデータの保持を行なうことができる。

2.2 データの性質

本研究では http プロトコルによるアクセスのみを対象とする。ftp などの他のプロトコルにおいてもキャッシングの効果はあると考えられるが、本稿では扱わない。

対象データの期間は、1995 年 11 月 7 日 13 時から 1995 年 11 月 28 日 13 時までである。アクセスの総数は 755633 であり、1 日平均 35982.5 アクセスがあることになる。

URL 単位のアクセス頻度分布を図 1 に示す。図中の直線は、後述の Zipf の法則 [5] を示す。また、図 2 にアクセス量と頻度、およびデータ総量の関係を示す。図 2 からは、URL 数では、ページデータサイズ 1k バイト付近が最も多いが、データ量では、同じく 10k バイト付近が最も寄与が大きいことがわかる。

図 1においてアクセス頻度が高いページは、教育系計算機用の WWW ホームページ関連である。教育系ワークステーションから mosaic を利用した場合にはホームページとしてまず参照され、mosaic はセッションを超えたキャッシングを行なわないためと考えられる。

キャッシュサイズが無制限の場合かつキャッシュデータの無効化が行なわれない場合のヒット率は 75.4% である。非常にアクセス頻度が高いページの影響を除いてアクセス回数が 100 回までのページを対象にした場合のヒット率は、68.7% である。この値は他の研究でのプロキシサーバの実測値 [4] と比べると大きい。これは、キャッシュデータの無効化の影響とトレースデータの大きさが関係していると考えられる。

ページの内容が更新された場合には、キャッシュデータを無効にして、サーバへのアクセスを行なうようにしないと最新の内容にアクセスできない。サーバでの内容更新に伴ってキャッシュからのパージが必要になる。しかし、ログデータからは、サーバでの内容更新やキャッシュデータの無効化の必要性を知ることができない。そこで、本稿では内容更新に伴うデータ無効化を行なわないとして解析を行なった。また、従来のプロキシサーバでの無効化パージは、一定時刻もしくは、容量制限を超えた時にまとめて行なわれるが、本稿の無効化 (パージ) は、キャッシュの制限を超えた時に行なわれる。

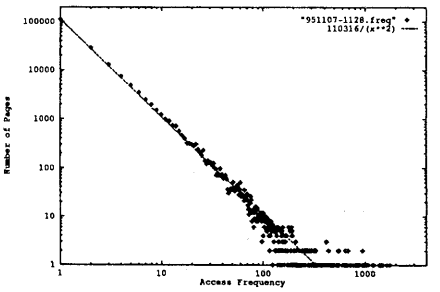

図 1: URL 単位のアクセス頻度分布

[3] では、ページのアクセス頻度は、Zipf の法則 [5] に従うと考えてもよいと述べられている。図 1 において、アクセス頻度が低い部分では Zipf の法則があてはまるが、アクセス頻度が高い部分では法則からのずれがかなりある。アクセス頻度が f であるページ数が $P(f)$ であるとすると f におけるアクセス回数は $A(f) = fP(f)$ となる。Zipf の法則が成り立つ場合には、$A(f) = M/f$ である。ここで M は定数である。変形すると $P(f) = M/f^2$ となる。図 1 には、$P(f)$ をプロットしてある。Zipf の法則に従ったアクセスがあり、キャッシュ容量に制限がなく、データの無効化が存

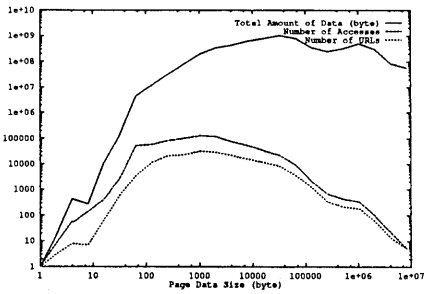

図 2: ページデータサイズに対するアクセス量、URL 数およびデータ総量との関係。ページデータサイズを 2 の巾乗で区間に分割して、その区間ごとに集計してある

在しないとした場合のヒット率 $\gamma(F)$ は、

$$\gamma(F) = \frac{H(F)}{N(F)} = \frac{\sum_{f=1}^{F} \frac{f-1}{f^2}}{\sum_{f=1}^{F} \frac{1}{f}}$$

である。ここで、最大アクセス頻度を F、総アクセス数を $N(F)$、ヒット数を $H(F)$ とする。$\gamma(100) \approx 0.685$ であり、上記のログの解析結果とよく一致している。

3 キャッシュ置き換えアルゴリズム

ここでは、まず、WWW データアクセスに適したキャッシュ置き換えアルゴリズムとして世代型置き換えを提案する。その後、従来の基本的な置き換えアルゴリズムを適用した場合と世代型置き換えアルゴリズムを適用した場合を比較し、提案の置き換えアルゴリズムの評価を行なう。

3.1 世代型キャッシング

プロキシサーバキャッシュのヒット率は、CPU の命令・データキャッシュに比べるとあまり高くない。前述のように、キャッシュサイズ無制限かつデータの無効化が行なわれない場合でも 80%以下である。CPU の命令・データキャッシュでは、90% 以上のヒット率が得られるのに比べると低いといってよい。

図 1のデータの性質を見ると 1 度しかアクセスされていないページがかなりあることがわかる。1 度しかアクセスされないエントリをキャッシュに残して置くことは無駄である。キャッシュの大きさを制限した場合には、実効的なキャッシュの大きさが減少することになるのでヒット率の低下につながる。

そこで、これを改善するために将来のアクセスの可能性に応じてキャッシュでの置き換えられやすさを変更するアルゴリズムを提案する。この方法を世代型キャッシングと呼ぶことにする。あまりアクセスが見込まれないエントリを早く置き換え対象になるようにする。1 度しかアクセスされていないものはアクセスされる見込みが低いと考える。

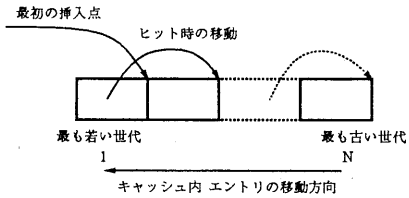

図 3: 世代型キャッシュの概念図

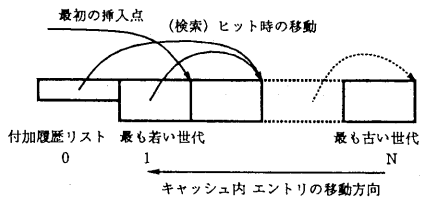

図 4: 付加履歴を利用する世代型キャッシュの概念図

世代型キャッシュの概念図を図 3に示す。世代の移行はかならずしも 1 世代単位で行なう必要はない。複数世代を飛び越すことがあってもよい。図 3では、1 世代単位の場合を示してある。最も若い世代は、大き過ぎると実効的キャッシュサイズの減少につながる。一方、最も若い世代が小さすぎると、実際には複数回のアクセスがあっても、より古い世代への移行が行なわれず、ヒット率の向上に役立たない。したがって、若い世代は、2 度以上のアクセスがある場合のアクセス間隔以上かつ十分に小さいことが望ましいと考えられる。世代の個数は任意であるが、本稿では、もっとも単純な場合として 2 世代に分割した場合を示す。

また、付加履歴を利用する世代型キャッシュの概念図を図 4 に示す。世代型キャッシュに、データの内容を持たないでエントリ情報のみを持つ付加履歴を追加したものが、付加履歴を利用する世代型キャッシュである。付加履歴中の検索でエントリが見つかってもデータは持たないのでキャッシュとしてはミスになるが、エントリの挿入点 (世代) が、付加履歴中で見つからなかった場合とで異なる。

3.2 置き換えアルゴリズム

本稿で結果を示すキャッシュ置き換えアルゴリズムを説明する。また、今後の説明に利用する略称を示す。

FIFO First-In First-Out アルゴリズム。ハードウェアキャッシュなど機構を単純にしたい場合に利用されている。エントリ数がキャッシュの大きさを超えた際に置換えが行なわれる。

LRU Least Recently Used アルゴリズム。キャッシュの置換えによく利用されている。

OPT 最適アルゴリズム。参照が最も遠い将来のエントリから置換えを行なう。あらかじめ、参照がすべてわかっていなければならず、実用的な意味はないが、これ以上よくすることはできないという限界を示す。

gp2 2 世代の分割をする世代型アルゴリズム。

gp2a 2 世代の分割をする世代型アルゴリズム。付加履歴を利用する。

上記のアルゴリズムでは、キャッシュ量の制限として URL 数のみを考える。しかし、ディスク容量などのデータ量の制限も考える必要がある。キャッシュの大きさをデータ総量で決める FIFO、LRU、OPT をそれぞれ FIFO-s、LRU-s、OPT-s で表す。また、データ総量によってキャッシュの大きさを決める場合の gp2 に相当するアルゴリズムを sgr2 とする。ただし、sgr2 における世代の分割点は、有効キャッシュエントリ数の割合で決める。

3.3 URL あたりヒット率

URL あたりのヒット率は、アクセスレイテンシとの関係が深い。ヒット率が高まるほど、ユーザの平均アクセスレイテンシが短縮される。

総アクセス URL 数が N であり、ヒットした URL 数が H である場合の URL あたりのヒット率を H/N と定義する。基本アルゴリズム、gp2、gp2a の場合のヒット率を、それぞれ、図 5、図 6、図 7 に示す。gp2 の場合は、LRU と比較して 3 ポイント程度のヒット率向上が得られることがわかる。また、gp2a の場合は、LRU と比較して 3.5 ポイント程度のヒット率向上が得られる。付加履歴を利用した gp2a の方が付加履歴を利用しない gp2 よりもヒット率を向上させることができる。

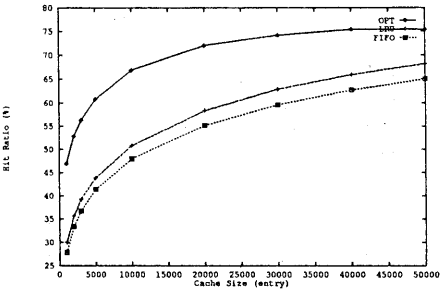

図 5: 基本アルゴリズムの場合のキャッシュサイズと URL ヒット率の関係

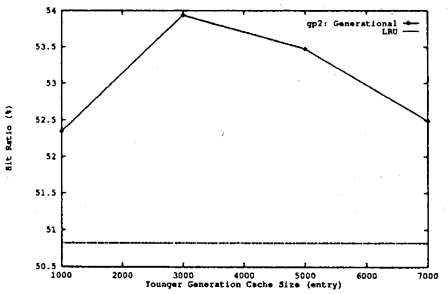

図 6: キャッシュサイズが 10000 エントリの単純分割 gp2 の場合の URL ヒット率

3.4 データ量あたりヒット率

データ量あたりのヒット率は、ネットワークへの負荷との関係が深い。ヒット率が高まるほど、外部との通信量が減少する。総アクセスデータ量を D とし、ヒットしたページのデータ量の合計を T とする。データ量あたりのヒット率は、T/D と定義する。

データ総量制限時の基本的アルゴリズムのヒット率を図 8 に示す。また、データ総量を 100MB に制限した際の世代型アルゴリズム sgr2 のヒット率をそれぞれ図 9 に示す。図中の URL、Amount は、それぞれ、URL あたりのヒット率、データ量あたりのヒット率を意味する。sgr2 においてのデータ量あたりのヒット率に 2 ポイント程度の効果があることがわかる。URL あたりのヒット率に対しては、gp2 と同様に 4 ポイント程度の効果である。100MB における効果の程度は、FIFO と LRU の差以上である。

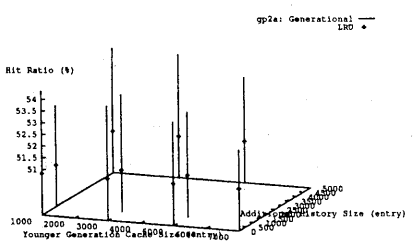

図 7: キャッシュサイズが 10000 エントリで付加履歴利用 gp2a の場合の URL ヒット率

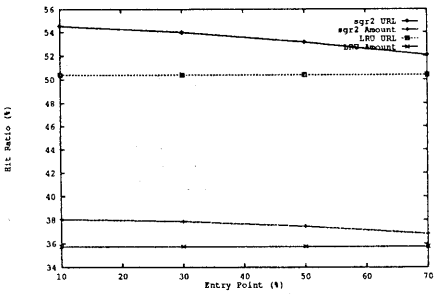

図 9: データ総量 100MB 制限時の世代型アルゴリズム sgr2 のヒット率

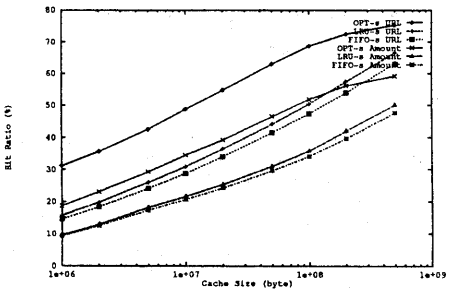

図 8: データ総量制限時の基本的アルゴリズムのヒット率

4 コネクションキャッシング

1 回のコネクションで複数のページを転送するプロトコルが提案されている [9]。あるページ内から参照されているページをまとめて処理するという方法である。提案の意図は、レイテンシを小さくすることである。

本研究では、提案されている方法とは異なるが、コネクション解放をページ単位でおこなわず、コネクションのキャッシングを行なう方法の評価を示す。

プロキシ側

プロキシ (クライアント) 側がサーバへのコネクションをキャッシングした場合の評価を図 10 に示す。「Proxy Side」と示してある。この評価もプロキシサーバのログを基にしている。この結果からは、コネクションキャッシングの効果がかなり高いことがわかる。32 コネクションをキャッシングする場合には、LRU でも 92%以上のヒット率が得られている。コネクションヒット率は、従来の CPU の命令・データキャッシュのヒット率に相当する。URL ヒット率は、1 アドレスのヒット率に相当すると考えることができよう。

ただし、この解析では、1 サーバに対しては 1 コネクションのみを設定すると想定しており、このままでは、同一サーバに対して複数の要求が同時に行なわれた場合には、レイテンシの増加を招く。複数のコネクションを許す場合の検討は、今後の課題である。

サーバ側

必ずしも大規模サーバとはいえないが、電気通信大学情報処理センターの WWW サーバへのアクセスログを基にして、サーバ側がコネクションキャッシングを行なった場合の評価を図 10 に示す。「Server Side」と示してある。ログの期間は、1995 年 11 月 7 日 13 時から 1995 年 12 月 5 日 13 時までである。アクセス総数は、124263 であった。この結果からは、サーバ側においてもコネクションキャッシングの効果がかなり高いことがわかる。32 コネクションをキャッシングする場合には、LRU でも 88%以上のヒット率である。

電気通信大学情報処理センターの WWW サーバへのアクセスは学内からが多く、頻度も十分ではない。アクセス頻度が大きな大規模サーバのログに基づいた解析が望まれる。また、コネクションキャッシングを有効な機構として運用するためには、キャッシュの置き換えの際のコネクションの切断を適切に処理する必要があり、具体的なプロトコルを検討する必要がある。

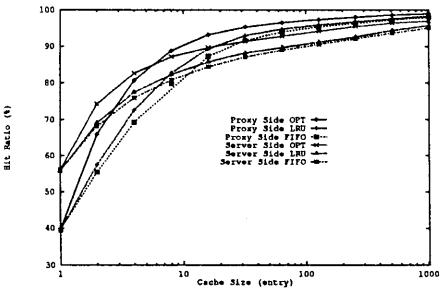

図 10: プロキシとサーバとでのコネクションキャッシュのヒット率

5 他の研究との関連

世代型キャッシュに関連した従来の研究との違いを説明する。

ガーベジコレクションを行なうシステムでは、多くのセルはすぐに死に、ある程度残っているものはかなり長い間残るということが経験則として知られている。この経験則を利用する世代別ガーベジコレクション [7] が提案され、実用に供されている。若い世代は頻繁にガーベジコレクションを行ない、古い世代は数少なくガーベジコレクションを行なうことによってガーベジコレクションのオーバーヘッドを少なくすることを目的としている。本稿のアルゴリズムのアイディアと似た点はある。しかし、本稿のアルゴリズムは、オーバーヘッドを小さくすることではなく、ヒット率を向上させることを目的にしている点が異なる。また、キャッシュのヒット率とガーベジコレクションの廃棄率とでは、統計的な性質も異なる。

キャッシュを分割するという方法は、UNIX のファイルバッファキャッシュにおいて先読みページの管理において一部利用されている。しかし、本稿のアルゴリズムは、先読みなどの情報に基づいておらず予測の基礎となる考え方が異なる。

6 おわりに

世代型キャッシングを提案し、LRU と比較して 3〜4 ポイント程度の効果があることがわかった。ただし、データ量あたりのヒット率の向上は、URL エントリ単位のヒット率の向上に比べて小さく、2 ポイント程度である。ヒット率の向上の程度は十分ではないので、より効果を高める方法をさらに検討する必要がある。

また、コネクションにキャッシュを導入すると、32 エントリで 88%以上のヒット率という大きな効果を期待できることがわかった。今後は具体的なプロトコルとその効果の評価を行なう予定である。

1 回のみの (もしくはアクセス間隔が非常に大きな) アクセスがかなりあり、良くアクセスされたものが将来もアクセスされる可能性が高いという傾向は、広いアクセス可能空間の一部にアクセスが行なわれる場合によく見られる傾向と考える。大規模なファイルシステムについても世代型置き換えアルゴリズムの効果があるかどうかを確かめる予定である。

謝辞

ログデータの入手の便宜を図っていただいた電気通信大学総合情報処理センターの各位、特に同センターの小林克志助手に感謝する。

参考文献

[1] Abrams, Marc, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams and Edward A. Fox, "Caching Proxies: Limitations and Potentials," Proc. of Fourth Int'l World Wide Web Conference, Dec. 1995.
(http://www.w3.org/pub/Conferences/WWW4/Papers/155/)

[2] Danzig, P.B., R.S. Hall and M.F. Schwartz, "A case for caching file objects inside internetworks," Proc. of ACM SIGCOMM '93, pp.239-248, Sept. 1993.

[3] Glassman, Steven, "A Caching relay for the World Wide Web," Proc. of 1st Int'l Conf. on WWW, Geneva, May 1994 ; also appeared in Computer Networks and ISDN Systems, Vol.27, pp.165-173, 1994.

[4] 一井 信吾, 中山 雅哉, "キャンパスネットワークにおける WWW キャッシングの効果", 情報処理学会 分散システム運用技術研究グループ資料 DSM-9505033, 1995 年 5 月 17 日.

[5] Knuth, Donald, *The Art of Computer Programming, Vol. 3. Sorting and Searching*, Addison-Wesley, 1973.

[6] Kwan T.T., R.E. MacGrath and Daniel Reed, "NCSA's World Wide Web Server: Design and Performance," IEEE Computer, Vol.28, No.11, pp.68–74, Nov. 1995.

[7] Lieberman, H. and C. Hewitt, "A Real-Time Garbage Collector Based on the Lifetimes of Objects," Comm. ACM, Vol.26, No.6, pp.419–429, June 1983.

[8] Luotonen, A. and K. Altis, "World-Wide Web Proxies," Proc. of 1st Int'l Conf. on WWW, Geneva, May 1994; also appeared in Computer Networks and ISDN Systems, Vol.27, pp.147-154, 1994.
(http://www1.cern.ch/PapersWWW94/luotonen.ps)

[9] Padmanabhan, V.N. and J.C. Mogul, "Improving HTTP Latency," Proc. Second Int'l WWW Conf., pp.995-1005, 1994.
(http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/mogul/HTTPLatency.html)

[10] Pitkow, J. E. and M. M. Recker, "A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns," Proc. of 2nd Int'l WWW Conf., pp.1039-1046, 1994.
(http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/pitkow/caching.html)