ナップザック問題の適用による WWW キャッシュアルゴリズムの設計手法

村本英司*, 山井成良†, 滝根哲哉‡, 村上孝三*

*大阪大学大学院工学研究科 情報システム工学専攻
〒 565-0871 大阪府吹田市山田丘 2-1
電話番号：06-879-7804　FAX 番号：06-879-7760
E-mail：muramoto@ise.eng.osaka-u.ac.jp
E-mail：murakami@ise.eng.osaka-u.ac.jp

†岡山大学 総合情報処理センター
〒 700-8530 岡山県岡山市津島中 3 丁目 1 番 1 号
電話番号：086-251-7238　FAX 番号：086-251-7244
E-mail：yamai@cc.okayama-u.ac.jp

‡ 京都大学大学院情報学研究科 数理工学専攻
〒 606-8501 京都府左京区吉田本町
電話番号：075-753-4758　FAX 番号：075-753-4756
E-mail：takine@kuamp.kyoto-u.ac.jp

内容梗概

インターネット上のトラヒックの大部分を占める WWW に対してキャッシュを設置し，サーベへのアクセス数，外部ネットワークのトラヒック量，クライアントへの応答時間などの改善を図る方法が一般的である．本稿では WWW キャッシュの性能に大きな影響を与えるキャッシュアルゴリズムを対象としている．ドキュメントの置換え問題がナップザック問題に帰着することに着目し，準最適なキャッシュアルゴリズムをキャッシュの目的に応じて設計するための手法を提案する．トレースデータ駆動型シミュレーションによる性能評価の結果，本手法より設計されたアルゴリズムは他のものより改善の割合が高く，その有効性が確認された．

A design method of WWW caching algorithm via knapsack problems

Eiji Muramoto*, Nariyoshi Yamai†, Tetsuya Takine‡, Koso Murakami*

*Department of Information Systems Engineering,
Graduate School of Engineering, Osaka University
Osaka, 565-0871, Japan
Phone：06-879-7804　FAX：06-879-7760

†Computer Center, Okayama University
Okayama, 700-8530, Japan
Phone：086-251-7238　FAX：086-251-7244

‡Department of Applied Mathematics and Physics,Graduate School of Informatics, Kyoto University
Kyoto, 606-8501, Japan
Phone：075-753-4758　FAX：075-753-4756

Abstract

World Wide Web (WWW) traffic is one of the most resource consuming applications on the Internet. The WWW caching is now widely used to reduce the load on Web servers, network traffic and average latency in fetching Web documents. This paper considers caching algorithms which can significantly affect the cache performance. We first show that various WWW caching algorithms can be formulated as knapsack problems. Based on this, we propose a method to design suboptimal caching algorithms. Through trace-driven simulation we confirm that designed algorithms perform better than other algorithms in the literature.

1 はじめに

近年，民間ネットワークプロバイダの出現やWWWなどの新しいアプリケーションの登場により，インターネット上のトラヒックの増加やクライアントへの応答時間の悪化が問題となっている．この問題の解決手法として，インターネット上のトラヒックの大部分を占めるWWWに対してキャッシュを設置し，外部ネットワークに転送されるアクセス数やトラヒック量の削減あるいはクライアントから見た応答時間の短縮を図る方法が最も一般的である．現実にもsquid[6]，apache[7]などのキャッシュ機能を持つソフトウェアが多くのサイトで導入されている．

キャッシュの空き容量が少なくなったとき，どのドキュメントをキャッシュ内に残し，どのドキュメントを削除するかを決定するアルゴリズム（以下，キャッシュアルゴリズム）はWWWキャッシュの性能に大きな影響を与える．本研究はナップザック問題がWWWキャッシュに適用できることを明らかにし，これを用いて準最適なキャッシュアルゴリズムを設計するための手法を提案する．

以下では，第2章ではWWWキャッシュの構成とその性能指標を説明する．第3章では従来のキャッシュアルゴリズムを紹介し，その問題点を示す．第4章ではアルゴリズム設計手法を提案し，その手法より各指標を改善するアルゴリズムを設計する．さらに第5章で設計されたアルゴリズムの性能評価の概要と結果を示す．最後に第6章でまとめと今後の課題を示す．

2 WWW キャッシュ

WWWキャッシュはネットワークの各地点に導入することが可能であるが，図1のようにクライアントと同一のドメイン内にプロキシとして配置され，複数のクライアントからの通信をサーバに中継する場合が多い．

キャッシュはドキュメントを一時記憶し，クライアントから同じドキュメントに対するアクセスがあった場合は記憶していたドキュメントのコピーをクライアントに伝送する．これにより，サーバとプロキシの間の通信が不要となり，外部サーバへのアクセス数，外部ネットワークのトラヒック量の削減，クライアントへの応答時間などが改善される

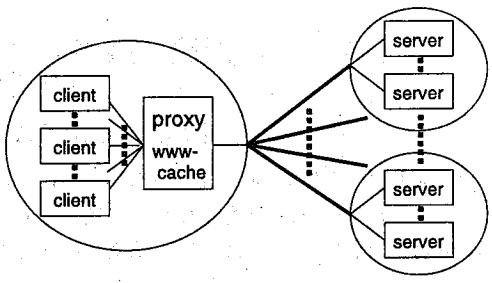

図1: ネットワーク構成

キャッシュの性能評価には一般に次の指標が用いられている．

ヒット率 (Hit Rate) アクセスされたドキュメントの総数に占めるキャッシュヒットしたドキュメントの割合を表す．この値が高いほど外部サーバへのアクセス回数が減少したことになる．

バイトヒット率 (Byte Hit Rate) 総伝送バイト数に対するキャッシュヒットしたドキュメントの総バイト数の割合を表す．この値が高いほど，外部ネットワークのトラヒックが減少したことになる．

平均応答時間 (Average Latency) ドキュメントをダウンロードするまでのクライアントの待ち時間を表す．前の2つと異なり，クライアントからのシステムの評価基準である．

3 従来のキャッシュアルゴリズム

実際のシステムではキャッシュ容量は有限であるため，アクセスされたすべてのドキュメントをキャッシュ内に残すことはできない．このため，キャッシュアルゴリズムが必要となる．

WWWキャッシュの性能を改善するため，現在まで多くのキャッシュアルゴリズムが提案されてきた．これらは大きく2種類に分類できる．1つは計算機システム上での主記憶管理に用いられるページ置換えアルゴリズムをWWWキャッシュにそのまま適用したものである．代表的なアルゴリズム

として，LRU(Least-Recently-Used)，LFU(Least-Frequency-Used) が知られている．LRU は最後にアクセスされてから最も長い時間の経過したドキュメントを，LFU はアクセス回数が最も少ないドキュメントをそれぞれキャッシュの中から置き換える．

一方，もう1つの分類は，主記憶管理には存在しない WWW の性質を考慮したアルゴリズム [1,2,3,4,5] で，代表的なものとして SIZE[1]，MIX[4] などが挙げられる．SIZE はサイズの小さなドキュメントを優先的にキャッシュに残すアルゴリズムで，キャッシュ内のドキュメント数を増やしてヒット率を高くすることを目的としている．MIX は平均応答時間の短縮を目的としたアルゴリズムであり，キャッシュ内のドキュメントの中で次の式 (1) の値が小さいものから順に置き換える．

$$\frac{(lat_j)^{r_1} * (nref_j)^{r_2}}{(tref_j)^{r_3} * (size_j)^{r_4}} \qquad (1)$$

lat_j：あるドキュメント j の最新の伝送時間
$nref_j$：格納されてからのアクセス回数
$tref_j$：最後のアクセスからの経過時間
$size_j$：サイズ
r_1, r_2, r_3, r_4：可変パラメータ

しかし，これらのアルゴリズムはいずれも経験則に基づくものであり，キャッシュの性能をどの程度改善するか明確でない．例えば，計算機システムの主記憶管理と WWW キャッシュ管理はサイズや外部からのアクセス速度がドキュメントに依存するなどの面で異なるため，ページ置換えアルゴリズムがそのまま WWW キャッシュにも適するかどうか不明である．また，SIZE や MIX はヒット率や平均応答時間をある程度改善するが，ドキュメントの選択基準に理論的裏付けがなく最適かどうか明らかでない．

4 アルゴリズム設計手法の提案

各指標について最適なキャッシュアルゴリズムを求めるには，ドキュメントの選択がキャッシュの性能に与える影響を理論的に明らかにする必要がある．そこで本研究では，まずキャッシュの性能を最適化するためにキャッシュに残すべきドキュメントの組み合わせを求める問題（以下，ドキュメント置換え問題）を定式化し，この問題がナップザック問題に帰着することを示す．次に，ナップザック問題の近似解法をもとに 2 章で示した各指標について準最適なキャッシュアルゴリズムの設計手法を示す．

4.1 ドキュメント置換え問題の定式化

まず，定式化のための仮定と表記を示す．

ドキュメントの総数を N 個，クライアントは過去の履歴と独立してアクセスを行うと仮定する．図 1 の構成において，各サーバに格納されている全てのドキュメントを対象として，各ドキュメントを $D_j (j = 1, \ldots, N)$ と表すことにする．また，D_j のサイズを s_j，クライアントが次に D_j をアクセスする確率を p_j と表す．

プロキシは容量 S (byte) のキャッシュを持ち，アクセスされたドキュメント D_j がキャッシュ内に存在する場合 (キャッシュヒット) にはクライアントに D_j のコピーを伝送し，また D_j がキャッシュに存在しない場合 (キャッシュミス) にはクライアントの代わりにサーバに D_j を要求し，そのコピーをキャッシュ内に格納しながらクライアントに中継する．キャッシュヒット時のプロキシからクライアントへの D_j の平均伝送速度を b_j (byte/sec)，キャッシュミス時のサーバからクライアントへの D_j の平均伝送速度を B_j (byte/sec) と表す．また，ドキュメント D_j がキャッシュに格納されているかどうかを表す変数として X_j を導入し，キャッシュに格納されている場合には $X_j = 1$，そうでない場合には $X_j = 0$ とする．

次にキャッシュに残したドキュメント D_j に対してクライアントからアクセスがあった場合に得られる利益 c_j について考える．この利益は指標によって異なり，ヒット率の場合は 1 回キャッシュヒットすると外部サーバへのアクセスが 1 回減少するので利益 c_j は 1(回) となる．バイトヒット率の場合は 1 回キャッシュヒットすると外部ネットワークのトラヒック量が s_j(bytes) 減少するので利益 c_j は s_j(bytes) である．平均応答時間の場合はキャッシュミス時の値 s_j/B_j(sec) からキャッシュヒット時の値 s_j/b_j(sec) に短縮されるため，その差 $\left(\frac{s_j}{B_j} - \frac{s_j}{b_j}\right)$(sec) が利益 c_j となる．なお，利益の期待値はドキュメント D_j のアクセス確率を考慮して $p_j c_j$ と表せる．

ドキュメント置換え問題はサイズ s_j の合計がキャッシュ容量 S 以下という条件下で利益 c_j の期待値 $p_j c_j$ の合計を最大にする $X_1, \ldots, X_N$ の組み合わせを見つける問題として (2) のように定式化できる.

$$\begin{aligned} &\text{find } \{X_1, \ldots, X_N\} \text{ such that} \\ &\text{maximize } \sum_{j=1}^{N} p_j c_j \\ &\text{subject to } \sum_{j=1}^{N} X_j s_j \leq S,\ X_j = 0 \text{ or } 1 \end{aligned} \tag{2}$$

式 (2) は $p_j c_j$ を価値として，その合計を最大にする $X_1, \ldots, X_N$ の組み合わせを容量 S の制約条件下で求める問題であり，ナップザック問題であることがわかる.

4.2 近似解法によるアルゴリズムの設計

ナップザック問題は NP 完全であり最適解を求めるのは困難である．このため，近似解法により準最適解を求める．ナップザック問題にはさまざまな近似解法 [8] が知られており，その一つに欲張り法 [8] がある．この欲張り法では単位サイズ当たりの価値 $p_j c_j / s_j$ の大きい順に，サイズ s_j の合計が S 以下という条件が満たされる限り $X_j = 1$ とし，残りを $X_j = 0$ とすることにより準最適解を得る.

したがって，ヒット率の場合は p_j / s_j の大きい順に，バイトヒット率の場合は p_j の大きい順に，平均応答時間の場合は $p_j \left(\frac{1}{B_j} - \frac{1}{b_j} \right)$ の大きい順に，容量制限に達するまでドキュメントをキャッシュに残していくアルゴリズムがそれぞれの指標に対する準最適解を得るアルゴリズムとなる.

4.3 パラメータの推定方法

設計したアルゴリズムにおける p_j, B_j, b_j の値は実際のシステムでは不明であり，過去のアクセス履歴など入手可能な情報をもとに推定する必要がある.

本稿では p_j をその時点までの $\frac{D_j\text{へのアクセス数}}{\text{全アクセス数}}$ で評価する．これは提案した設計手法よりバイトヒット率を改善するアルゴリズムが p_j の大きい順にキャッシュに残す方法と見い出されたこと，ならびに 5 章の性能評価においてバイトヒット率を最も改善しているアルゴリズムがアクセス回数を規準とした LFU であることが根拠である.

また B_j, b_j は URL に含まれるサーバ単位で集計を行い，その平均値を用いる．これはサーバ単位で集計を行うとドキュメント単位の場合に比べて，保持すべきデータが少なくなり，さらに平均値を求めるためのサンプル数が多く信頼性が高くなると思われるからである.

5 性能評価

提案したアルゴリズム設計手法の有効性を確認するために 4 章で設計されたアルゴリズムの性能を評価する必要がある．本研究ではトレースデータ駆動型シミュレーションにより性能評価を行った．本章ではシミュレーションの概要と結果について述べる．なお，以下では 4 章で設計した平均応答時間を改善するアルゴリズムは MAL(Minimum Average Latency)，ヒット率を改善するアルゴリズムは MHR(Maximum Hit Rate) と表記する.

5.1 使用したトレースデータの概要

まず，シミュレーションで用いたトレースデータの作成方法とその性質について説明する．我々はトレースデータとして大阪大学情報処理教育センターの squid サーバで記録されたアクセスログを使用した．観測期間は平成 10 年 5 月 31 日（日）0:00 ～ 6 月 8 日（月）21:00 で総アクセス数は約 190 万，総伝送バイト数は約 12 Gbyte である.

設計したアルゴリズムを評価するために必要な情報は各ドキュメントのアクセス時刻，URL，サイズである．また，MAL ではさらにキャッシュヒット時とキャッシュミス時の両方の伝送時間が必要である．しかし，ログにはキャッシュヒット時あるいはキャッシュミス時の伝送時間のどちらかの記録がなく，これらを測定しなければならない．そこで我々はアクセスログの伝送時間は使用せず，記録された各 URL からキャッシュ不可能なドキュメント（cgi ドキュメントなど）と削除などにより正常に伝送されなかったドキュメントを除いた総数 1,124,872 の URL についてサーバへ実際にアクセスを行ってヒット時とミス時の両方の伝送時間を求めた．なお，この測定のために Webjamma[2] を用いた．表

表 1: トレースデータの統計的性質

アクセス総数	1,124,872
ドキュメント総数	477,435
サーバ総数	13,507
アクセス一回のドキュメント総数	348,871(72.2%)
キャッシュが無いときの平均応答時間	526(msec)
ドキュメントの平均サイズ	7,353(byte)
キャッシュからクライアントへの平均伝送速度	67.8(kbyte/sec)
サーバからクライアントへの平均伝送速度	11.1(kbyte/sec)

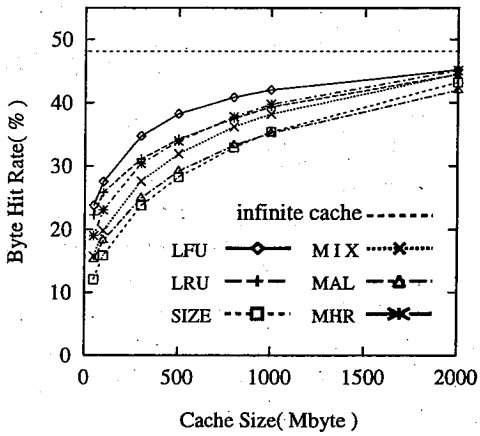

図 2: 種々のキャッシュサイズでの各アルゴリズムのバイトヒット率

1 に収集したデータの統計的性質を示す.

5.2 シミュレーション結果

4.1 節で示したトレースデータを用いて，設計したアルゴリズム MAL，MHR の性能評価を行った．また，比較の対象として LRU，LFU，SIZE，MIX の 4 種類のアルゴリズムについてもシミュレーションを行った．MIX の可変パラメータは [4] と同様に $r_1 = 0.1, r_2 = r_3 = r_4 = 1$ に設定した．

はじめにキャッシュサイズを無限大に設定してプログラムを実行した．その結果，キャッシュに格納されたドキュメントの総バイト数は 4.29 (Gbyte) となった．また，ヒット率の最大値は $HR_{max} = 57.6(\%)$ バイトヒット率の最大値は $BHR_{max} = 48.2(\%)$ 平均応答時間の最小値は $Res_{min} = 383$ (msec) となった．

次にキャッシュサイズを変化させて各アルゴリズムのバイトヒット率，ヒット率，平均応答時間を測定した．シミュレーションで用いたキャッシュサイズは 50Mbyte ～ 2000Mbyte の 7 通りである．また，squid と同様にドキュメントの累計蓄積量が高位水準に達したときにキャッシュアルゴリズムを起動させ，低位水準になるまでドキュメントを除去し続ける方法を採用した．これは置換えに要する時間を短縮するためである．本シミュレーションでは squid の標準設定値を用いて，高位水準をキャッシュ容量の 90%，低位水準を 75% とした．

まず，図 2 にバイトヒット率のシミュレーション結果を示す [1]．4.5 節でも述べたが，LFU が全てのキャッシュサイズにおいてバイトヒット率を最も改善している．したがって，アクセス確率 p_j にその時点までの $\frac{D_j\text{へのアクセス数}}{\text{全アクセス数}}$ の値を用いて，MAL，MHR を実行した．

さらに図 3 に各アルゴリズムのヒット率，図 4 に各アルゴリズムの平均応答時間の改善の割合のシミュレーション結果を示す．図 4 はキャッシュがない場合の平均応答時間 526(msec) を 100% とした場合の，それぞれのキャッシュサイズでの各アルゴリズムの改善の割合を表している．この値が小さいほど平均応答時間が短縮されたことになる．図 3 ではヒット率を改善するアルゴリズムとして設計された MHR が，図 4 では平均応答時間を改善するアルゴリズムとして設計された MAL が全てのキャッシュサイズにおいて最も優れた性能を示している．これにより，我々の提案したアルゴリズム設計手法の有効性を確認できる．また，MHR，MAL はともにキャッシュサイズが 100 ～500Mbyte の範囲のときに特に優れた値を示している．キャッシュサイズが 1000 ～2000Mbyte のときにアルゴリズムによる差が比較的現れにくいのは，ドキュメン

[1] 図中の infinite cache と表された線はキャッシュサイズが無限大の場合の値である．図 3，図 4 も同様

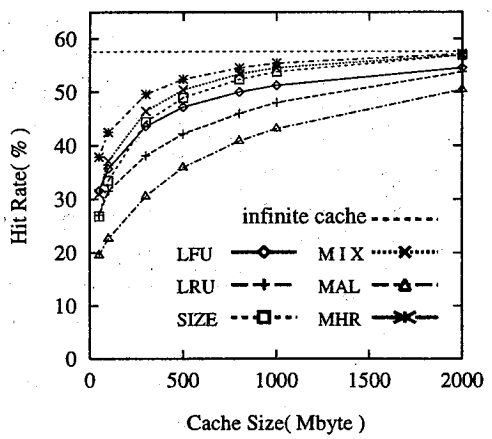

図 3: 種々のキャッシュサイズでの各アルゴリズムのヒット率

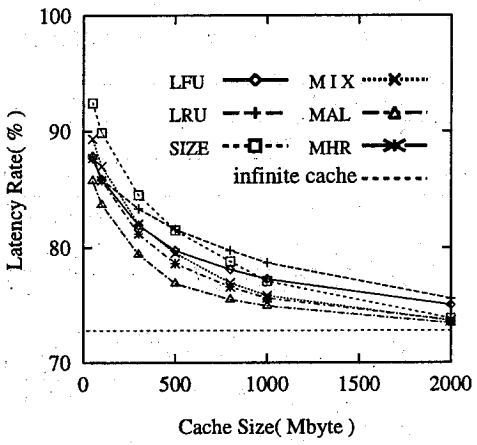

図 4: 種々のキャッシュサイズでの各アルゴリズムの平均応答時間の改善の割合

トの大部分がキャッシュに格納され，ヒットとミスが分かれる場合が少ないことが理由と考えられる．また，キャッシュサイズが 100Mbyte 以下の場合は，キャッシュサイズが小さすぎるためヒットするドキュメントが少なく，そのために他のアルゴリズムと差が出にくいと考えられる．

6 まとめ

本稿では WWW のドキュメント置換え問題がナップザック問題に帰着することに着目して，これに基づいたキャッシュアルゴリズムの設計手法を提案した．次に各指標についてキャッシュの性能を改善するアルゴリズムをナップザック問題の近似解法をもとに設計した．さらに，トレースデータ駆動型シミュレーションを行い，これらのアルゴリズムの性能を評価した．この結果，提案した設計手法の有効性を確認できた．

今後の課題としてはドキュメントのアクセス確率やネットワーク帯域のより良い推定方法の考案が挙げられる．

参考文献

[1] S.Williams, M.Abrams, C.R.Standridge, G.Abdulla, and E.A.Fox, "Removal Policies in Network Caches for World-Wide Web Documents", Proc. ACM SIGCOMM'96, pp.293-305, 1996.

[2] R.P.Wooster and M.Abrams, "Proxy Caching that Estimates Page Load Delays", Proc. 6th World Wide Web Conference, pp.325-334, 1997.

[3] P.Cao and S.Irani, "Cost-Aware WWW Proxy Caching Algorithms" Proc. 1997 USENIX Symposium on Internet Technology and Systems, pp.193-206, 1997.

[4] Z.Liu, P.Nain, and N.Niclausse,"A new efficient caching policy for the world wide web", Proc. Workshop on Internet Server Performance, 1998.

[5] L.Rizzo and L.Vicisano, "Replacement policies for a proxy cache", UCL-CS Research Note RN/98/13, 1998.

[6] "Squid Internet Object Cache", http://www.nlanr.net/Squid/

[7] "Apache Project", http://www.apache.org/

[8] N.Christofides, A.Mingozzi, P.toth, and C.Sandi, "Combinational Optimization", Wiley, 1978.