# サイト単位のアクセス負荷変動を考慮したWWWキャッシュ置き換えアルゴリズムの提案

長田智和† 神里大† 谷口祐治‡ 玉城史朗†
†琉球大学理工学研究科
§琉球大学総合情報処理センター

## 概 要

近年，WWW情報サービスの普及よって，ネットワークやサーバの負荷が増大し，WWWサービス応答時間の遅延が問題となっている．この問題を解決するため，我々はWWWキャッシュサーバにおける新たなキャッシュ置き換えアルゴリズム "*GTSFD*" を提案し，性能評価によってWWWサービス応答時間の改善を確認した．一方，*GTSFD* では，逐次計算されるデータサイズ，アクセス頻度，スループットの平均値を基にキャッシュ優先度を求めているが，アクセス負荷変動が優先度の決定に反映されていないため，サイトや時間帯によってキャッシュオブジェクトに公平な優先度を与えることができない．そこで，*GTSFD* にサイトや時間帯ごとのアクセス負荷変動を考慮する仕組みを付加することを提案する．これによって，キャッシュ性能のばらつきが少ないキャッシュ置き換えアルゴリズム (*GTSFD+*) が実現可能であると考えられる．

# A WWW Cache Replacement Algorithm utilizing Access Load Effect for Every Site

Tomokazu NAGATA† Masaru KAMIZATO† Yuji TANIGUCHI‡ Shiro TAMAKI†
† Graduate School of Engineering and Science, University of the Ryukyus
‡ Computing and Networking Center, University of the Ryukyus

## Abstract

In recent years, information services using WWW spread rapidly and delay of WWW service response time is caused by increase of network traffic and server load. In order to solve this problem, we proposed a new cache replacement algorithm "*GTSFD*" in a WWW cache server. On the other hand, the cache priority is calculated serially by the average of the data size, access frequency and throughput in *GTSFD*. Since the access load effect is not reflect in a priority decision, a fair priority can not be given to cache object. In order to solve this defect, we propose adding the structure in consideration of the load effect for every site and access time to *GTSFD*. By this method, we considered realizing a cache replacement algorithm (*GTSFD+*) with little dispersion in the cache performance by the access load effect.

## 1 はじめに

近年，World Wide Web(以下，WWW) を用いたインターネット上での情報サービスが急速に普及し，ネットワークやサーバの負荷増大によってWWWサービス応答時間の遅延が問題となっている．また，ネットワークやサーバの負荷増大は，複数のユーザが同一ページを閲覧することでさらに助長されている．これらの問題を解決するため，WWWキャッシュサーバ (図1) が広く利用されている．一方，従来のWWWキャッシュサーバで用いられているページデータの管理 (キャッシュへの保存や削除) のためのキャッシュ置

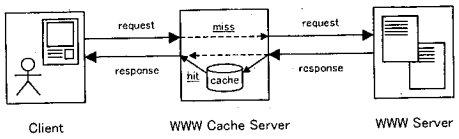


図 1: WWW キャッシュシステム

き換えアルゴリズム [1, 2, 3, 4] は，アクセス頻度やデータサイズから主にヒット率やバイトヒット率の改善を目的とした手法 [5, 6] は多いが，クライアントが要求したページデータを受信するのに要するサービス応答時間を明確に改善する有効な手法はない．そこで，我々は WWW キャッシュサーバにおける新たなキャッシュ置き換えアルゴリズム "*GTSFD*"[7] を提案し，性能評価によって WWW サービス応答時間の改善を確認した．一方，*GTSFD* では，逐次計算されるデータサイズ，アクセス頻度，スループットの平均値を基にキャッシュ優先度を求めているが，アクセス負荷変動がキャッシュ優先度の決定に反映されていないため，サイトや時間帯によってキャッシュオブジェクトに公平な優先度を与えることができない．そこで，*GTSFD* にサイトや時間帯ごとのアクセス負荷変動を考慮する仕組みを付加することを提案する．これによって，キャッシュ性能のばらつきが少ないキャッシュ置き換えアルゴリズムが実現可能であると考えられる．

# 2 GTSFD(Greedy Triple, Size Frequency Distance)

## 2.1 動作概要

我々は，ヒット率およびバイトヒット率を改善しつつ，特に WWW サービス応答時間の改善を目的としたキャッシュ置き換えアルゴリズム "*GTSFD*" を提案した．*GTSFD* の動作概要は次の通りである．すなわち，図 2 においてアクセス頻度およびデータサイズが大きく，スループットが小さいページデータほどキャッシュ優先度 (ページデータをより長期間キャッシュするための評価値) を高くし (A)，その逆の場合は低くする (B)．また，1 度キャッシュしたページデータが再びリクエストされた場合は，アクセス頻度が高いページデータほどキャッシュ優先度を高くし (A)，その逆の場合は低くする (B)．一方，アクセス頻度，データサイズ，スループットは各々次元 (単位) が異なるため，そのままでは対等な価値として評価することはできない．そこで，各評価値のスケーリングを行い，対等な価値として扱えるようにする．

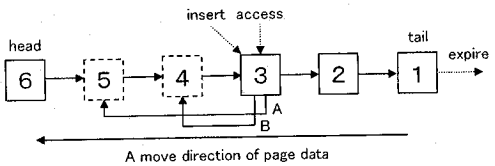


図 2: *GTSFD* 動作概要

以上によって，ヒット率およびバイトヒット率を改善しつつ，サービス応答時間を改善を実現する．

## 2.2 優先度評価式の導出

従来のキャッシュ置き換えアルゴリズムでは，キャッシュ優先度を決定する評価指標としてアクセス頻度やデータサイズを用いるのが一般的であった．*GTSFD* ではこれらに加えて，スループットを新たな評価指標として採用する．すなわち，スループットを (転送データサイズ／サービス応答時間) と定義し，その逆数，つまり，単位データサイズあたりのサービス応答時間が長いほどキャッシュ優先度を高くする．ここで，データサイズ，アクセス頻度，1/スループットの 3 つの評価指標に相関がないことを確認していることから，これらの評価指標は互いに独立と見なせる．*GTSFD* では，3 つの評価指標で張られる空間を 3 次元の直交空間 (ユークリッド空間) と定義し，原点からの距離 (ユークリッド距離) を該当ページデータのキャッシュ優先度とする．また，各評価指標は次元 (単位) が異なるため評価指標同士に対等な価値を与えるスケーリングを行う必要がある．さらに，各々の出現分布 (分布関数：ヒストグラム) は，特定の分布関数に従うと仮定することは困難であるため，それらの統計量 (平均や分散など) を用いた近似分布関数を与えることはできない．そこで，各評価指標の平均値を基に指数関数的な価値に基づく距離を導入する．以上のことを，データサイズを例にとって以下で説明する．

まず，データサイズを $\alpha$，データサイズの平均を $\overline{\alpha}$ として，原点からの距離 (データサイズ評価値：$\alpha_{val}$) を次式で与える．

$$\text{データサイズ：}\quad \alpha_{val} = 1 - exp(-\frac{\alpha}{\overline{\alpha}}) \qquad (1)$$

この関数は，一般に一次遅れ系の応答関数と呼ばれ，

$\alpha$ が小さいほど原点からの距離が短く，逆に，$\alpha$ が大きいほど原点からの距離が長くなる．ここでは，$\overline{\alpha}$ を時定数と見なしており，$\alpha = \overline{\alpha}$ であるならば原点からの距離 (つまりデータサイズ評価値) が 0.632 となる．すなわち，平均値による正規化操作を行っていることを意味する．さらに，他の 2 つの評価指標 (アクセス頻度評価値 : $\beta_{val}$，1/スループット評価値 : $\gamma_{val}$) についても同様に，

$$\text{アクセス頻度：}\quad \beta_{val} = 1 - exp(-\frac{\beta}{\overline{\beta}}) \qquad (2)$$

$$1/\text{スループット：}\quad \gamma_{val} = 1 - exp(-\frac{\gamma}{\overline{\gamma}}) \qquad (3)$$

と表すことができる．式 (1)〜式 (3) より，$GTSFD$ におけるキャッシュ優先度 ($GTSFD_{val}$) は 3 変数のユークリッドノルムを用いて次式で定義する．(分母 $\sqrt{3}$ はキャッシュ優先度を正規化するための係数である)

キャッシュ優先度：

$$GTSFD_{val} = \sqrt{\frac{\alpha_{val}^2 + \beta_{val}^2 + \gamma_{val}^2}{3}} \qquad (4)$$

## 2.3 評価実験

$GTSFD$ は，WWW キャッシュサーバとして広く利用されている Squid[8] のソースコードを変更することで実装し，実ネットワークで運用した．また，比較のため LRU(Least Recently Used), LFUDA(Least Frequently Used with Dynamic Aging), GDSF (Greedy-Dual Size-Frequenccy) の 3 つの従来手法によるデータ取得も行い，各手法におけるデータ取得ではキャッシュ置き換えアルゴリズムを変更した以外の設定 (MemoryCache:512MB, DiskCache:6GB) は全て同一とした．さらに，WWW キャッシュサーバを運用したネットワークは，バックボーン帯域が 15Mbps，接続端末数が約 4,200 台，ネットワーク利用者数が約 8,600 人，データ取得期間中の 1 日あたりの WWW キャッシュサーバへのリクエスト数は約 100 万リクエストであり，ゲートウェイルータによる透過プロキシによってリクエストを収集した．また，各手法のデータ取得において 450 万リクエスト取得し，120±5 万リクエストの範囲でディスクキャッシュあふれを確認した．

## 2.4 評価・考察

まず，スループット，サービス応答時間，バイトヒット率において，450 万リクエスト時点で $GTSFD$ が最も良好な結果が得られた．また，ヒット率については従来手法及び $GTSFD$ ともにほぼ同等の結果であった．ここで，データ取得期間中の全リクエストに対するページデータの平均サイズを見てみると，450 万リクエスト時点で GDSF が最も大きく，次いで LFUDA, LRU, GTSFD の順となった．さらに，キャッシュヒットしたページデータの平均サイズは，GTSFD が最も大きく，次いで LRU, LFUDA, GDSF の順であった．

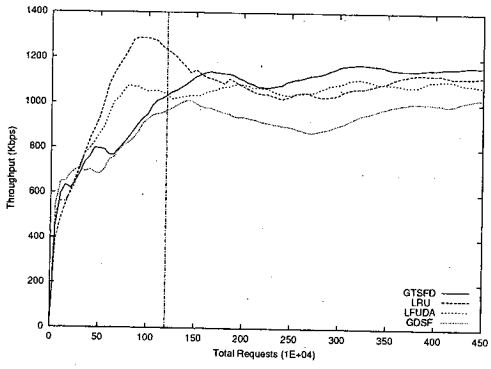


図 3: スループット

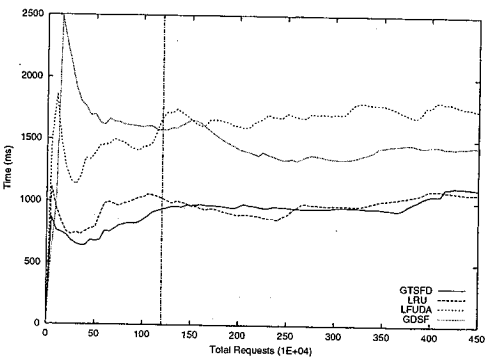


図 4: サービス応答時間

以上の結果から，$GTSFD$ ではスループットが改善されることによってサービス応答時間も改善されている．スループットが改善された要因は，評価式によって，サービス応答時間が長いページオブジェクトが優先的にキャッシュされているためである．また，大きなデータサイズのキャッシュオブジェクトを優先的にキャッシュすることによって，バイトヒット率も改善されている．ここで，スループットおよびサービス応答時間において，$GTSFD$ が LRU とほぼ同等の結果

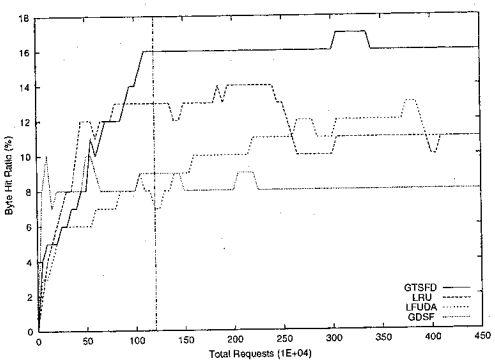


図 5: バイトヒット率

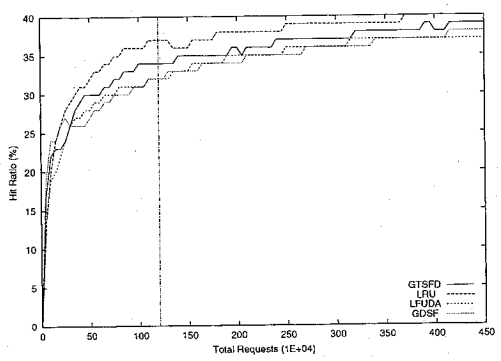


図 6: ヒット率

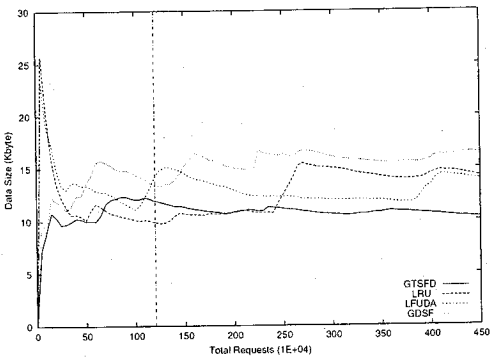


図 7: 平均データサイズ (全リクエスト)

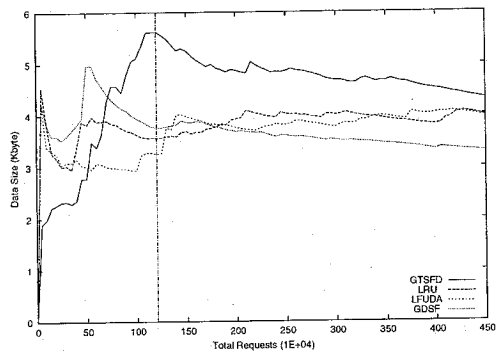


図 8: 平均データサイズ (ヒットリクエスト)

になっている要因は，LRU のデータ取得期間中の総リクエストにおける平均データサイズが，$GTSFD$ に対して約 35%大きかったことが影響していると考えられる．すなわち，LRU 運用時はより大きなページオブジェクトがキャッシュされやすい条件であったため良好な結果が得られている．もし，総リクエストの平均データサイズが $GTSFD$ と同程度であった場合は，$GTSFD$ は LRU に対してもより明確に優位性が確認できたと考えられる．

## 2.5 問題点

前節において，$GTSFD$ の性能評価からその有効性を確認した．しかし，以下のような問題点を指摘することができる．すなわち，$GTSFD$ では逐次計算されるデータサイズ，アクセス頻度，スループットの平均値を基にキャッシュ優先度を決定していることから，サイトや時間帯によってキャッシュオブジェクトに公平な優先度を与えることができないと考えられる．図 9〜図 12 に 4 日間 (H14/10/11-H14/10/14) の $GTSFD$ におけるヒット率，バイトヒット率，スループット，サービス応答時間の 1 日の変動を示す．

これらの結果によると，時間帯によってキャッシュ性能に大きなばらつきがあることが確認できる．この原因は，例えばスループットを例に取ると次のように説明することができる．すなわち，ある時間におけるスループット値が，それまでの平均値よりも高かった場合，1/スループット優先度が低く与えられる．ところが，同じページオブジェクトを別の時間にリクエストするとスループットが平均値よりも低い場合も考えられ，その場合は先に与えた優先度は適切なものではないということになる．さらに，アクセス負荷変動はサイトによっても異なるため，逐次計算される平均値を使ったキャッシュ優先度の決定は，サイトや時間帯

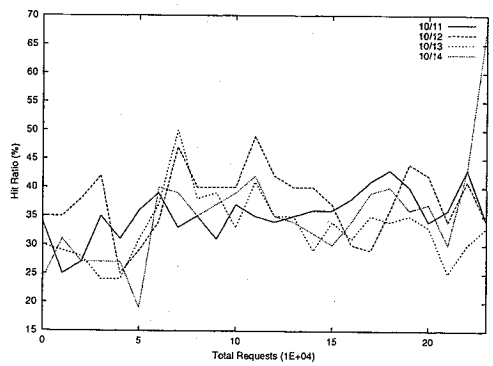


図 9: ヒット率

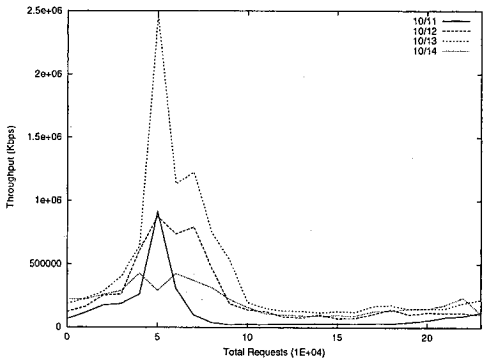


図 11: スループット

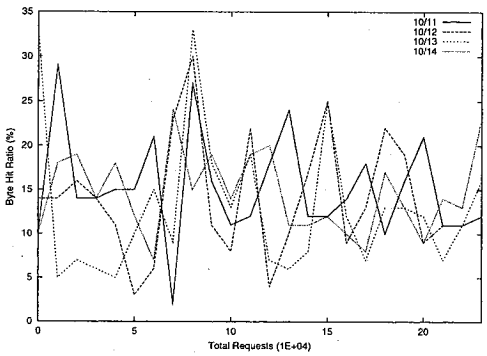


図 10: バイトヒット率

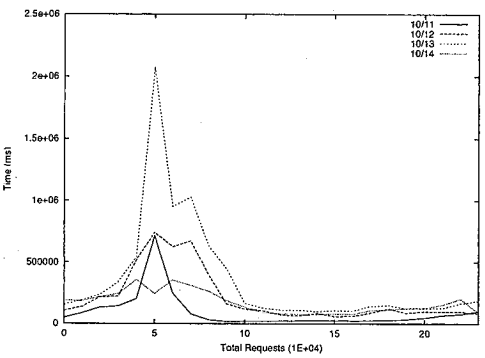


図 12: サービス応答時間

ごとにキャッシュオブジェクトに公平なキャッシュ優先度を与えているとは言えない.

このことから，我々は，*GTSFD* にサイトや時間帯ごとのアクセス負荷変動を考慮する仕組みを付加し，キャッシュ性能のばらつきが少ない新たなキャッシュ置き換えアルゴリズムを提案する.

# 3 GTSFD+(Greedy Triple, Size Frequency Distance plus)

## 3.1 動作概要

*GTSFD+* では，*GTSFD* でリクエストごとに逐次計算していたデータサイズ，アクセス頻度，スループットの平均を，1 日を 1 時間ごとに 24 区間に分け，1 時間ごとの平均を用いることで優先度を決定する. また，時間ごとの平均はサイトごとに独立とする. これによって，サイトや時間帯ごとのアクセス負荷変動に依存しない，公平なキャッシュ優先度を求めることができると考えられる.

次節では以上の仕組みを追加したキャッシュ優先度評価式を導出する.

## 3.2 優先度評価式の導出

*GTSFD+* では，サイトごとに 1 日 (1 時間ごとに 24 サンプル) の履歴を保持し，1 時間単位で各評価値の平均を求め，それを基にキャッシュ優先度を決定する. すなわち，*GTSFD* における優先度評価式に，サイト単位のカテゴリ分けと時間軸が追加されることになる. 以下に，データサイズ，アクセス頻度，1/スループットおよびキャッシュ優先度評価式を示す.

$$\text{データサイズ：}\quad \alpha_{val_{site_t}} = 1 - exp(-\frac{\alpha_{site_t}}{\bar{\alpha}_{site_t}}) \quad (5)$$

アクセス頻度： $\beta_{val_{site_t}} = 1 - exp(-\frac{\beta_{site_t}}{\overline{\beta_{site_t}}})$ (6)

1/スループット： $\gamma_{val_{site_t}} = 1 - exp(-\frac{\gamma_{site_t}}{\overline{\gamma_{site_t}}})$ (7)

キャッシュ優先度：

$$GTSFD_{val_{site_t}} = \sqrt{\frac{\alpha^2_{val_{site_t}} + \beta^2_{val_{site_t}} + \gamma^2_{val_{site_t}}}{3}} \quad (8)$$

なお，実装において用いる各評価値の平均を求める逐次式は，例えばデータサイズの場合は，

$$\overline{\alpha}_{x_k} = \frac{1}{k}\{(k-1)\overline{\alpha}_{x_{k-1}} + \alpha_{x_k}\} \ (k = \text{データ数}) \quad (9)$$

となる．(他の評価値も同様に求められる)

### 3.3 履歴保存領域の見積り

$GTSFD+$ において，サイトごとに履歴を保持するために必要な領域を計算する．$GTSFD+$ では，サイトごとに 1 日 24 サンプルのデータサイズ，アクセス頻度，スループットの平均値を保持するが，$GTSFD$ での 4000 万リクエストにおよぶ評価実験の結果，出現サイト数は 12 万サイトで増加が飽和した．これら 12 万サイト全てについて履歴を保持するとすると，必要な領域 ($hist.space$) は，

$hist.space =$

[サイト数]*[サンプル数]*[評価指標] = 約 $70MB$ (10)

となる．

低価格化，大容量化が進むメモリおよびストレージを考慮すれば，必ずしも過剰な領域ではないと言える．

## 4 まとめ

### 4.1 結論

本論文では，まず，我々が提案したスループット評価指標の導入によるサービス応答時間を改善した WWW キャッシュ置き換えアルゴリズムについて述べ，性能評価からその有効性を指摘した．次に，$GTSFD$ の問題点を指摘し，サイトや時間帯ごとのアクセス負荷変動を考慮する必要性について WWW アクセス分析から言及した．さらに，その解決策として新たに $GTSFD+$ を提案し，その効果を予見した．

### 4.2 今後の課題

本論文では $GTSFD$ をベースとしてサイトや時間帯ごとのキャッシュ性能のばらつきを軽減するキャッシュ置き換えアルゴリズム $GTSFD+$ を提案した．現在，この $GTSFD+$ を Squid へ実装し，実ネットワークにおいて運用中である．今後は，運用データから性能比較を行い，$GTSFD$ と比較してサイトや時間帯ごとにキャッシュ性能にばらつきが少ない結果が得られるか確認を行っていく．

## 参考文献


[1] Abrams, M., Standridge, C.R., Abdulla, G.,Williams, S. and Fox, E.A.: Caching Proxies: Limitations and Potentials, 4th International World Wide Web Conference (1995).

[2] 大澤範高，早野文孝，弓場敏嗣，箱崎勝也: 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，情報処理学会マルチメディアと分散処理研究会報告，Vol. 96-DPS-74，No. 33，pp. 191–196 (1996).

[3] Scheuermann, P., Shim, J. and Vingralek, R.: A Cache for Delay-Conscious Caching of Web Documents, 6th International World Wide Web Conference, pp. 725–734 (1997).

[4] Wooster, R.P. and Abrams, M.: Proxy Caching that Estimates Page load Delays, 6th International World Wide Web Conference, pp. 325–334 (1997).

[5] Pei, C. and Sandy, I.: Cost-Aware WWW Proxy Caching Algorithms, Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (1997).

[6] Dilley, J., Arlitt, M. and Perret, S.: Enhancement and Validation of Squid's Cache Replacement Policy, HP Labs Technical Reports, HPL-1999-69 (1999).

[7] 長田智和，神里大，谷口祐治，玉城史朗: スループット評価指標を導入した WWW キャッシュ置き換えアルゴリズムの提案と評価，銃砲処理学会論文誌 (投稿中).

[8] Squid: http://www.squid-cache.org/