連続メディアデータ対応のウェブキャッシングアルゴリズム

佐々木 盛朗† 田中淳裕† 立川 江介†

概要

本稿では、オブジェクトの種類毎に異なる方法で優先度を割り当て、キャッシュ記憶領域を動的に割り当てることで、複数の異なる種類のオブジェクトを効率的にキャッシュするキャッシングアルゴリズム、Prioritized Group Caching (PGC)を提案、評価する。PGCは、連続メディアと従来型のウェブオブジェクトをキャッシュするのに有効である。評価では、実トレースと合成トレースを使ったシミュレーションを行い、どちらのトレースを用いた場合にも、PGC は既存のアルゴリズムよりも高いヒット率を示した。また、合成トレースを用いた場合、キャッシュサイズを 32MB とした時に既存のアルゴリズムに対して 4.9%～33%のページヒット率の向上(相対比)がみられた。

A Web Caching Algorithm for Multiple Object Types

Shigero Sasaki† Atsuhiro Tanaka† Kosuke Tatsukawa†

ABSTRACT

We propose a novel Web caching algorithm for multiple object types that separately gives priority to each type of object and also coordinates the share of cache storage that each type is given. This algorithm can meet the growing demand for caching both continuous media data and Web objects. The effectiveness of the algorithm was examined by simulation with real and synthetic traces. In the real trace, the algorithm outperforms other well-known caching algorithms. In the synthetic traces, the algorithm improves page hit rate by 4.9—33% compared to other existing algorithms when the cache size is 32MB.

1. はじめに

ウェブキャッシングでは、リクエストされたウェブオブジェクトをネットワーク的に近くにあるキャッシュサーバにキャッシュしておき、以降のそのオブジェクトへのリクエストにはキャッシュサーバが応える。これによって、ネットワーク的に遠くにあるサーバからのトラフィックを削減できるため、レスポンスタイムの短縮、ネットワーク帯域の節約ができる。これらの効果はキャッシュヒットによってもたらされるので、過去にリクエストされたオブジェクトを全てキャッシュするのが望ましい。しかし、記憶容量の制限から、キャッシングアルゴリズムによって、キャッシュするオブジェクトの選択が行われる。したがって、適切なキャッシングアルゴリズムを用いてキャッシュヒット率を高めることが重要である。

その一方で、近年のオブジェクトの多様化によって、異なる特性をもったオブジェクトを同時にキャッシュする必要が生じている。主な例としては、音声やビデオといった連続メディアオブジェクト(CO)と、従来型のウェブオブジェクトである非連続メディアオブジェクト(NCO)を同時にキャッシュする、といったことが挙げられる。CO は、巨大なサイズとオブジェクトの先頭からシーケンシャルにアクセスされるという特性を持つ。そのため、例えば、動画の再生開始から 30 秒後と 20 秒後の画像が同時にリクエストされている、といったことが起こる。この2つのリクエストは 10 秒の差で同じデータにアクセスするため、先行リクエストが参照したデータをキャッシュすれば、ほぼ確実に後続リクエストによるキャッシュヒットが起こる。

Interval Caching (IC)[6][7] は、このリクエスト同士の参照地点の間隔(interval)を利用して、CO を効率的にキャッシュする手法であるものの、interval が存在しないと適用できないため、NCO のキャッシングには不向きである。これに対して、NCO 向けのキャッシングアルゴリズムは、CO の特性を考慮できない。

そこで本稿では、複数の異なる種類のオブジェクトを効率的にキャッシュするアルゴリズム、Prioritized Group Caching (PGC)を提案する。PGC は、オブジェクトの種類別にキャッシングアルゴリズムを適用し、種類毎にキャッシュするための記憶領域を動的に割り当てる。

†NEC インターネットシステム研究所 (Internet Systems Research Laboratories, NEC Corporation)

2. 関連研究

最適なキャッシングアルゴリズムは、最も遠い将来に参照されるオブジェクトをパージするアルゴリズムである[3]。ところが未来の参照が既知でない限り、最適アルゴリズムは実現できないため、Least Recently Used (LRU)や Least Frequently Used (LFU)といったアルゴリズムで未来の参照を予測することになる。

ウェブオブジェクトへの参照には、i 番目に参照頻度の高いオブジェクトへの参照頻度は $1/i^{\alpha}$ に比例するという参照頻度の局所性がある[2]。トレースデータの分析から、α の値は 0.6 から 0.8 の間であることが[4]で示されている。また、ウェブオブジェクトの参照には時間的局所性もあり、前回の参照時刻から現在時刻までの時間を t として、オブジェクトの参照確率は、$1/t$ に比例することが知られている[5]。

参照回数に基づく LFU 系のキャッシングアルゴリズムは、キャッシュポリューションと呼ばれる問題を抱えている。これは、過去に頻繁にリクエストされていたが、最近はあまりリクエストされていないページがキャッシュに存在し続けて、ヒット率を低下させる現象である。Least Frequently Used with Dynamic Aging (LFUDA) [8]は、エイジと呼ばれる時間に対して単調に増加する数を、参照されたオブジェクトの参照回数に加算して、そのオブジェクトの優先度としている。そのため、参照回数の持つ重み／価値が時間の経過とともに相対的に減少していき、キャッシュポリューションを防止できる。LFUDA では、参照回数を用いることで参照頻度の局所性を利用し、またエイジを用いて時間的な要因を考慮できるので LRU や LFU よりも高いキャッシュヒット率を得ることができる。

Interval Caching (IC)[6][7] は CO 向けのキャッシングアルゴリズムである。IC は、オブジェクトの参照確定部のみをキャッシュする。例えば、10MB のオブジェクトの先頭から 1MB 目のデータへの参照と 2MB 目のデータへの参照が起きている場合、オブジェクトの 1MB—2MB のデータは参照確定部(interval)となる。そして、先行するリクエストによって参照されたデータをキャッシュしておけば、後続のリクエストによって起こる参照はキャッシュヒットとなる。IC では、小さな参照確定部から優先的にキャッシュすることで、少ないリソースで多くのキャッシュヒットを得る。ところが、参照確定部がなければデータがキャッシュされないところが IC の問題点であり、NCO のキャッシングに使われない理由である。CO と NCO を同時にキャッシュする方法としては、CO と NCO とに対してそれぞれ静的にリソースを割り当てる方法[1]もあるが、CO と NCO への参照の割合は動的に変化するため、両者への参照頻度を考慮して動的にリソースを割り当てる方が好ましい。

3. Prioritized Group Caching

PGC は、複数の異なる種類のオブジェクトをキャッシュすることを目的としたアルゴリズムである。PGC では、オブジェクトの種類別に適用するキャッシングアルゴリズムを設定でき、どの種類のオブジェクトがどれ位の記憶領域を占有するかを、動的に変更できる。したがって、PGC は CO と NCO のキャッシングに適用できる。PGC の用途はこれだけに限られないが、以降では、CO と NCO のディスク—メモリ間のキャッシングを対象として考える。

3.1 キャッシュ単位と優先度決定のための情報について

多くのウェブキャッシングアルゴリズムでは、オブジェクト単位にキャッシュやパージを行うが、CO と NCO を対象とした PGC においては、キャッシュ単位はページ(4KB のデータ)とする。

PGC では、個々のオブジェクトにキャッシュ優先度決定のための情報を持たせ、これをメモリ中に置く。PGC ではページ毎にキャッシュやパージを行うため、ページ毎にキャッシュ管理のための情報を持つのが自然であるが、メモリ節約の観点から、ある一つのオブジェクトに属する異なるページ間で共有する情報は、ページ毎には管理せずにオブジェクト側で一括して管理する。このような情報としては、オブジェクトが無効(Expire)になる時刻や、オブジェクトの前回参照時刻や参照回数などが考えられる。本稿では、参照回数のみを用いた PGC について述べる。メモリ中のページに対しては、ページ間の優先順位を判断するために必要な優先度を、それぞれのページに対して付加する。この優先度の算出・決定方法は3.3節で述べる。

ディスク中のオブジェクトが持つ優先度決定のための情報は、参照回数のみである。他の情報を持たないのはメモリ節約のためであり、情報を持たない分、より多くのページをメモリにキャッシュするためである。

3.2 ページの類別

CO と NCO を対象とした PGC では、ページを二つのグループに分ける。一つは predictable group であり、このグループのページは、次回参照時刻が予測できるページである。もう一つ

は unpredictable group であり、次回参照時刻が予測できないページからなる。Predictable なページは、同一オブジェクトの異なるページがリクエストされた時の、リクエスト間のページである。図 1には、12 ページからなるオブジェクトに対して二つのリクエストがあって、一方のリクエストに対しては既に 1—7 ページ目を返していて、もう一方のリクエストには 1—4 ページ目を返している場合を図示した。5—7 ページ目は、先行するリクエストによって参照されたときにキャッシュしておけば、後続のリクエストによって参照されるので predictable なページとし、1—4 ページ目には後続のリクエストがなく、キャッシュしても参照されるとは限らないため、unpredictable なページとする。

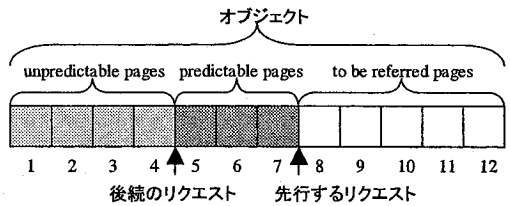

図 1. Predictable pages と unpredictable pages

ページが属するグループは実行時に変化する。Unpredictable なページは後続のリクエストが発生した瞬間に predictable になり、predictable なページは後続リクエストを満たした瞬間に unpredictable になる。

3.3 優先度割り当て

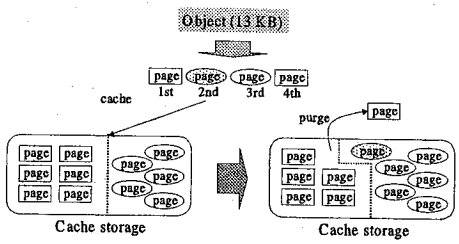

図 2. PGC の動作

図 2に、PGC の動作を示す。図は、到着した 13KB のオブジェクトが四つのページに分割されて、それぞれキャッシュされる場合の PGC の動作を表している。また、楕円で表される二、三番目のページは、predictable group に属するページであり、長方形で表される一、四番目のページは、unpredictable group に属するページである。図示されているのは、二番目のページをキャッシュし、predictable と unpredictable の各グループ内で最低の優先度を持つページをパージ候補として選択し、その結果 unpredictable なページをパージする動作である。

PGC は、到着したページが所属するグループを決定し、そのグループに適用されるアルゴリズムに従って到着ページに優先度を割り当てる。PGC は predictable group に属するページには次回予測参照時刻を優先度として与える。この時刻は後続のリクエストによってページが参照される時刻に等しいので、以下の式で次回予想参照時刻 *ntp* を与える(値が大きいと優先度は低い)。

$$ntp = ct + distance / ATR$$

ただし、*ct* は現在時刻を表し、*ATR* はページの平均転送速度を、*distance* は、後続のリクエストが参照しているページと優先度を割り当てるページの間にあるページ数を表す。例えば、図 1ではオブジェクトの 5 番目のページを後続リクエストが参照しているので、7 番目のページの優先度の計算で用いられる *distance* は 2 になる。この優先度は、*ATR* が一定のとき、IC と本質的に等価な優先度であり、IC では参照確定部全体(図 1ではページ 5, 6, 7)に対して優先度を割り当てるが、PGC では各ページに対して異なる優先度を割り当てる点に相違がある。Unpredictable なページには LFUDA を用いて優先度を割り当てる。LFUDA では、優先度として参照回数とエイジとの和を用いるキャッシュ管理手法である。エイジはパージが起きるたびに更新され、その値はパージされるオブジェクトの優先度(最低優先度)に等しく設定される。従って、新たに到着したページは必ずキャッシュされる。ページがパージされる度に、つまり時間が経つと、より大きい新たな最低優先度が定まるため、参照回数が多いページでも、長期間にわたって参照されなければパージされる。

PGC はグループにも優先度を割り当て、最も低い優先度を持つグループにおいて、最低の優先度を持つページをパージする。Predictable group の優先度は、predictable group に属するページの中で最も低い優先度に等しい。一方、グループ間で優先度比較を行うために、unpredictable group の優先度も時刻で割り当てる。unpredictable group の優先度 *ntu* を、

$$ntu = ct + C / f,$$

で与える。ただし、*ct* は現在時刻、*f* は unpredictable group に属するページで最も優先度が低いページの参照回数、*C* は経験的に定められる定数である。上記の式は、参照回数が多いページほど近い将来に参照されると予測して、頻繁に参照される unpredictable なページと、遠

い将来に参照される predictable なページのどちらをキャッシュに残すべきかの判断を行うことを目的とし、ntu が ntp 以下の時には predictable group からページをパージし、そうでなければ unpredictable group からページをパージする。表1に、PGC が割り当てる優先度を示す。ただし、$A(t)$は時刻 t における LFUDA のエイジである。

表 1. PGC における優先度割り当て(値が大きいほど優先度は小さい)

	predictable	unpredictable
page	$ct + distance/ATR$	$(f + A(t))^{-1}$
group	$ct + distance/ATR$	$ct + C/f$

PGC における動的リソース分配は、ページのキャッシュ、パージを通して行われる。例えば、到着した predictable なページをキャッシュしてキャッシュ中の unpredictable なページをパージすれば、自動的に predictable group が使用するメモリ領域が増え、unpredictable group が使用するメモリ領域が減るためである。

4. シミュレーションによる評価

この節では、CO と NCO を対象として、ディスクに記憶されたデータをメモリにキャッシュする場合の、PGC の性能をシミュレーションで評価する。シミュレーションでは、the National Laboratory for Applied Network Research (NLANR)[9]によって公開されていた実トレースと、CO への参照が増加した場合を想定して作成した合成トレースを用いた。

シミュレーションでは、実トレースでのオブジェクトへのリクエストを、そのオブジェクトを構成するページへのリクエストに変換して、ある時間間隔をおいて発行した。合成トレースもオブジェクトへのリクエストからなるので、同様にページリクエストに変換して評価に用いた。また、全ての評価対象のアルゴリズムはページリクエストを処理するように変更した。

4.1 評価指標

通常、キャッシングアルゴリズムの性能はキャッシュヒット率によって測定される。これはキャッシュ中のオブジェクトにヒットした数をオブジェクトへの全リクエスト数で割ることで得られる。しかし、オブジェクト単位のヒット率は PGC の評価指標として適当でないので、ページ単位のキャッシュヒット率であるページヒット率を評価指標として用いる。これは、キャッシュ中のページにヒットした数をページへの全リクエスト数で割ったものである。

さらに、predictable group に属するページ数の平均値として *pred* を導入する。時刻 t として、ページリクエストが発生する度に値が離散的に増加する時刻を考える。さらに、R を総ページリクエスト数、$p(t)$を時刻 t において存在する predictable group に属するページ数とすると、pred は以下の式で与えられる。

$$pred = \sum_{t=0}^{R} p(t)/R$$

また、predictable なページの数の分布を表す指標として *dist(k)* を導入する。*dist(k)* は、predictable なページが k 個以下しか存在しない確率を表し、*v(condition)*を *condition* が成り立つ時には1を、成り立たない時には0をとる関数として、以下の式で与えられる。

$$dist(k) = \sum_{t=0}^{R} v(p(t) < k)/R.$$

*dist(k)*は、i) 無限容量のキャッシュがあるとき、平均でどれ位の割合のページが predictable なページとしてキャッシュに存在するか、または、ii) 有限容量のキャッシュにおいて、最大で何ページの predictable なページをキャッシュし得るかを表す。ページヒット率、*pred, dist(k)*は次節で用いられる。

4.2 実トレースを用いたシミュレーション

実トレースとして使用したのは、NLANR が公開していたトレースで、2000年12月20日のトレースの、最初から約 100,000 リクエスト分であり、リクエストされたオブジェクトのバイト数は約 1GB である。これらのリクエストをページリクエストに変換すると、306,787 ページリクエストになり、229,467 種類のページがリクエストされた。またページサイズは 4KB とした。

NLANR のトレースには、リクエストによってキャッシュサーバに接続されたコネクションが閉じられた時刻 *timestamp* と クライアントソケットが accept されてから close されるまでの時間 *elapsed time* が記録されているので、以下の式でオブジェクトの *current* 番目のページへのリクエスト発行時刻 を計算した。ただし、*total* はオブジェクトを構成するページの数である。

$$timestamp - elapsedtime \times (total - current)/total$$

シミュレーションにおける LRU、perfect LFU (PLFU)[4]、LFUDA、PGC それぞれのキャッシングアルゴリズムのページヒット率を図 3に示す。ただし、perfect LFU はパージされたオブジ

エクトの参照回数も保持する LFU である。図 3 の横軸はキャッシュサイズ(MB)、縦軸はページヒット率(%)である。IC が評価対象になっていないのは、IC が unpredictable なページをキャッシュできないためである。

キャッシュサイズが小さい時に LRU のヒット率が最も低いのは、LRU がキャッシュサイズを越える時間的局所性を利用できないためである。一方、キャッシュポリューションはキャッシュサイズによらずに起こるので、キャッシュサイズが増加すると、PLFU と LRU のページヒット率が逆転する。

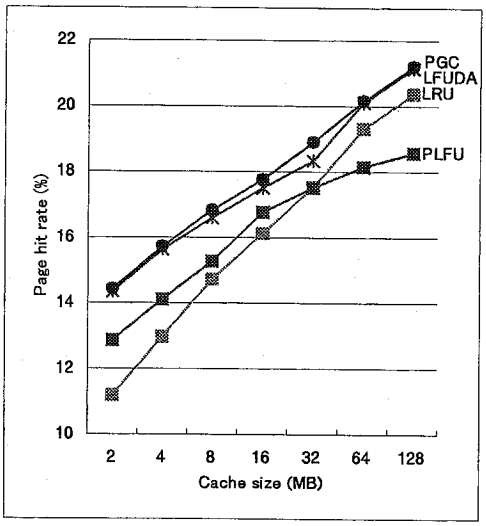

図 3. アルゴリズム別ページヒット率

LFUDA は PGC を除けば最も高いヒット率を示している。これは、エイジングによるキャッシュポリューションの抑制と、参照の局所性の利用による。IC と LFUDA をサブアルゴリズムとして使用する PGC は LFUDA よりも高いヒット率を示した。これは、少ないながらも実トレースから生成される predictable なページをキャッシングする手段を PGC が持っているためである。実トレースでは、平均で約 690 の predictable なページが存在した(*pred*≒690)。

4.3 合成トレースを用いたシミュレーション

この節では、将来における CO の増加を仮定して作成した合成トレースを用いて PGC の評価を行った。合成トレースは、典型的な NCO への参照に、CO への参照を追加して作成した。NCO への参照頻度は、4.2節で用いた実トレースと同様の Zipf-like 分布— i 番目に頻繁に参照されるオブジェクトは、$1/i^{0.778}$ に従う比で参照される—に従うことを仮定して作成した。また、NCO のサイズに関しては、平均値が 13KB である指数分布に従うとした。そして、ネットワークの帯域は 320 Kbps とし、オブジェクトへのリクエストは 0.5 秒間隔で発行されるとした。CO は 1MB のオブジェクトとし、CO の参照頻度は NCO と同じ分布に従うとした。そして、CO への参照が全リクエストの 0.5%, 0.75%, 1%, 1.5%, 2%, 3%, 5%, 10%を占めるように、CO へのリクエストを追加した。

オブジェクトの *current* 番目のページのリクエスト時刻は、

$$o + 0.1 * current / total,$$

で与えられる。ただし、o はオブジェクトのリクエスト時刻、0.1(4KB/320Kbps)は一ページ送信するのにかかる時間(秒)、*total* はオブジェクトを構成するページの数である。

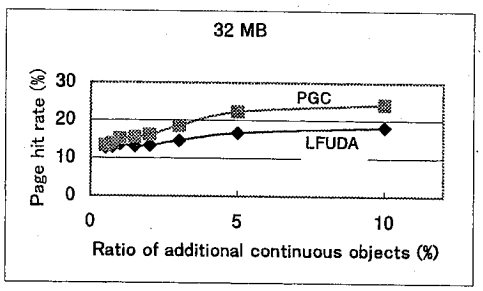

図 4. 合成トレースに対する PGC と LFUDA のページヒット率の比較(cache size = 32MB)

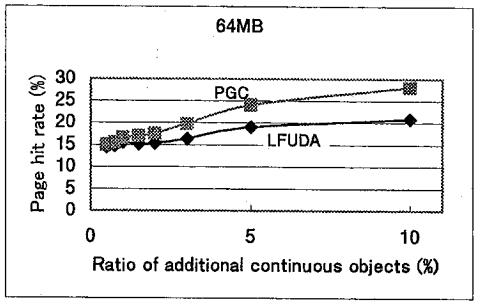

図 5. 合成トレースに対する PGC と LFUDA のページヒット率の比較(cache size = 64MB)

キャッシュサイズ 32MB と 64MB の時の PGC と LFUDA のページヒット率を図 4と図 5に示す。横軸は追加した CO へのリクエストの割合、縦軸はページヒット率である。PGC の LFUDA に対する相対的なページヒット率の向上は、32MB のキャッシュでは 4.9%—33%であり、64MB のキャッシュでは 3.3%—34%であった。

1%, 3%, 5%の CO へのリクエストの追加を行うと、各トレースに関して、*pred* = 2417, 4323, 5401 となる。表 2に、1%, 3%, 5%の CO へのリクエストを追加した合成トレースに対して pred 個の predictable なページがキャッシュサイズに占める割合を示す。PGC の LFUDA に対する優位性が、predictable なページが増えるほど顕著になるのが、図4、図5と表2から分かる。

表 **2. Predictable** なページがキャッシュサイズに占める割合

	Cache size	
Additional rate	32 MB	64 MB
+1%	29.5%	14.8%
+3%	52.8%	26.4%
+5%	65.9%	33.0%

4.4 Predictable group に属するページの数の分布

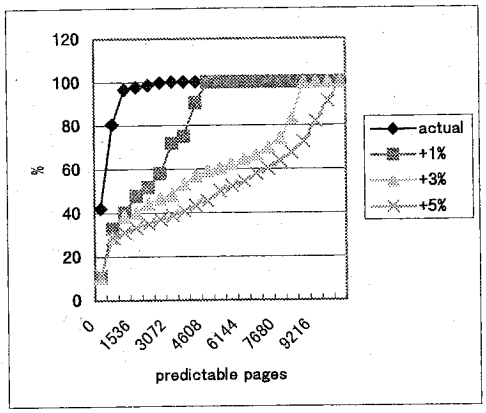

図 **6. Predictable** なページ数の分布 (**+*x*%**は ***x*%*** の **CO** を追加した合成負荷を表す)

4.2節と4.3節で示した PGC のページヒット率の高さの原因は二つある。一つは、同一オブジェクトへの連続した参照の性質を十分に利用していることであり、もう一つは、動的なリソース配分を行うことである。

図 6は、predictable なページがある数以下存在する確率を表す。横軸は predictable なページの数であり、縦軸は横軸の値 *k* に対する *dist(k)* である。例えば、3%(図の+3%)の CO を付加された合成トレースで predictable なページが 3072 ページ以下しか存在しない確率は 48.3%である (*dist*(3072) = 48.3%)。図 6において傾きがなだらかであるトレースほど predictable なページの数がとる値が等しい確率で広範囲に分布し、動的リソース割り当ての必要性が強くなり、PGC の有効性が顕著になる。

5. おわりに

本稿では、異なる種類のオブジェクトをキャッシュするためのキャッシングアルゴリズム、Prioritized Group Caching (PGC)を提案した。PGC は、主に従来型の非連続メディアデータと連続メディアデータをキャッシュすること目的としている。実トレースを用いたシミュレーションを行うことで、PGC と他のよく知られたアルゴリズムの比較を行い、PGC のページヒット率が最も高いことを示した。また、将来のトラフィックを予測して作成した合成トレースに対しては、キャッシュサイズを 32MB とした時に LFUDA に対する相対比で 4.9%から 33%のページヒット率の向上を示した。

ヒット率向上の理由は、PGC は異なる種類のオブジェクトに対して、それぞれ適当な既存のアルゴリズムで管理し、かつ、各種のオブジェクトのキャッシングに用いるリソース量を動的に調整することである。

6. 参考文献

[1] Almeida, J. M., Eager, D. L. and Vernon, M. K., A Hybrid Strategy for Streaming Media Files. In *Proc. of IS&T/SPIE Conference on Multimedia Computing and Networking 2001 (MMCN 2001)*, pages 200–212, January 2001.

[2] Almeida, V., Bestavros, A., Crovella, M. and Oliveira, A., Characterizing reference locality in the WWW. In *Proc. of IEEE International Conference in Parallel and Distributed Information Systems*, 92–103, 1996.

[3] Belady, L., A study of replacement algorithms for a virtual-storage computer. IBM System Journal, 5, pages 78–101, 1966.

[4] Breslau, L., Cao, P., Fan, L., Phillips, G. and Shenker, S., Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. of IEEE Infocom*, pages 126–134, 1999.

[5] Cao, P. and Irani, S., Cost-Aware WWW Proxy Caching Algorithms. In *Proc. of the USENIX Symposium on Internet Technologies and Systems*, pages 193–206, December 1997.

[6] Dan, A., Dias, D., Mukherjee, R., Sitaram, D., and Tewari, R., Buffering and Caching in Large-Scale Video Servers. In *Proc. of IEEE CompCon*, pages 217–224, 1995.

[7] Dan, A., Heights, Y. and Sitaram, D., Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads. In *Proc. of SPIE's conference on Multimedia Computing and Networking*, pages 344–351, January 1996.

[8] Dilley, J. and Arlitt, M., Improving Proxy Cache Performance: Analysis of Three Cache Replacement Policies. *IEEE Internet Computing*, 3(6), pages 44—50, 1999.

[9] The National Laboratory for Applied Network Research, http://www.nlanr.net