メモリベース・キャッシング代理サーバの実装
— アクセスピーク時における応答時間劣化の軽減手法 —

Implementaion of Memory Based Caching Proxy Server
— A Reduction Method of Degradation Response Time on an Access Peak —

梶田 朋己　中村 豊　知念 賢一　砂原 秀樹
Tomomi Kajita　Yutaka Nakamura　Ken-ichi Chinen　Hideki Sunahara

奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

概要

代理サーバの一日のアクセスには偏りがあり、アクセスの集中する時間帯 (ピーク) が存在する。ピークになると、キャッシング代理サーバの応答が非常に悪化する。

我々は、ピークにおいて、従来のキャッシング代理サーバの応答時間がどの程度劣化するのかについて調査を行った。そして、ピークにおける代理サーバの応答時間劣化の原因の一つがディスクアクセスにあると考えた。

そこで、本論文では、実際にメモリベース・キャッシング代理サーバを実装し、従来のシステムと比較することで、メモリベースにより応答時間劣化を軽減できることを示した。

1　はじめに

World-Wide Web(WWW) は、手軽に利用できる Internet のサービスとして、一般に広く利用されている。また、その扱い易さと柔軟な拡張性により、新たな市場としてさまざまな企業から注目されている。このような理由から WWW のトラフィックは今後も増加していくと考えられる。一方、トラフィックを軽減する技術として、WWW キャッシングが広く利用されている。WWW キャッシングとは、クライアントが参照した WWW オブジェクトを保持しておき、後続の要求に対して保持しているオブジェクトを返す技術である。WWW キャッシングにより、通信の発生を抑制し、また、クライアントへ即時に応答を返すことが可能となる。

WWW キャッシングには、クライアントで行うクライアントキャッシングと、代理サーバ上で行うキャッシング代理サーバがある。キャッシング代理サーバでは、複数のクライアント間でキャッシュを共有することができる。

キャッシング代理サーバを利用して WWW にアクセスする際に、サーバからの応答が非常に遅くなる時間帯がある。これは、キャッシング代理サーバへのアクセスが集中するからである。従って、アクセスが集中しても平常時と同じような応答性をもつキャッシング代理サーバが必要とされている。

キャッシング代理サーバのアクセス集中時の応答時間劣化は、ディスク I/O の待ち時間に大きく影響されると考えられる。そこで我々は、アクセス集中時の応答時間劣化を軽減手法として、メモリベース・キャッシング代理サーバの利用を提案する。

メモリベースのキャッシング代理サーバの提案は、すでにされている [1] が、実際に実装された例はない。そこで、実際にメモリベースによるキャッシング代理サーバの実装を行ない、実装した代理サーバを使って計測を行う。そして、計測により得られた結果を従来システムと比較しその有効

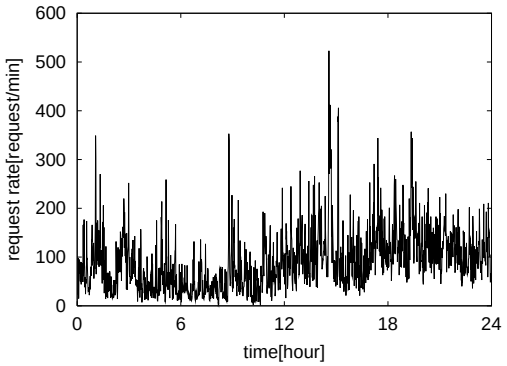

図 1: 奈良先端大の代理サーバの一日のアクセス

性を示す。

2 代理サーバアクセスのピークに関する議論

企業や大学などで利用されている代理サーバには、アクセスの傾向に時間的な偏りがある。特に休み時間などにアクセスが集中することはよく知られている。このようなアクセスの集中する時間帯に WWW を利用すると、キャッシング代理サーバからの応答が非常に悪くなる。本論文では、以後、一日を通してアクセスの集中する時間帯をピークと呼び、それ以外の時間帯をオフピークと呼ぶことにする。

一日のアクセスの動向をグラフにすると、アクセスが集中するピークがスパイク状にたびたび発生する (図 1)。このようなピークは、サイトの規模や利用者数によって大きく変化するので、一般化することは難しい。

しかし、ピークが発生しそうな時間帯は予測可能である。例えば、企業内で利用されている代理サーバであれば昼休みの時間帯や、勤務時間の終了直後に、ほぼ確実にピークがあらわれると予想される。

ピークの幅も、ピークの発生する時間帯から予測が可能である。昼休みに発生するピークを例に考えてみる。昼休みに発生するピークの原因は、仕事の手を休めて WWW を利用している人である。従って、代理サーバへのアクセスは、昼休みの終了と同時に減少する。このことから、ピークの幅は昼休みの長さに近い値になると考えられる。

最後にピークの高さについて考える。ピーク時のアクセス発生数は、オフピークの 3 倍程度になると言われている [2] 。

3 従来の代理サーバを用いた応答時間の測定

本章では、まず代表的な代理サーバのピークとオフピークの応答時間の差を測定する。測定を行う前に、測定に利用するクライアントのアクセスモデルと、性能評価のために利用する応答時間について議論する。次に、測定に使った環境について述べ、最後に結果を示す。次章で、結果に対する考察を行う。

3.1 ピークのモデル化

測定を行う前に、代理サーバにおけるピークのアクセスモデルを定める必要がある。

特定サイトの時間帯アクセスを計測することは可能である。しかし、時間帯アクセスを一般化することは困難である。また、このような複雑なアクセスを人為的に発生させることも非常に困難である。

そこで我々は、複数発生するピークの中の一つに注目し、ピークとその前後で応答時間にどの程度の差があらわれるのかを測定することにした。本論文では、容易に生成できる凸型のアクセスモデルを採用した。

2 章で述べたように、ピークにはオフピークの 3 倍程度のアクセスが発生すると言われている。そこで本実験では、多少の余裕ある性能を期待するために、オフピークの 4 倍のアクセスが発生するようにした。

リクエストの生成には、代理サーバのベンチマークソフト polygraph [6] を利用した。polygraph は、図 2 のようにクライアントとサーバで構成されたシステムで、クライアントとサーバの

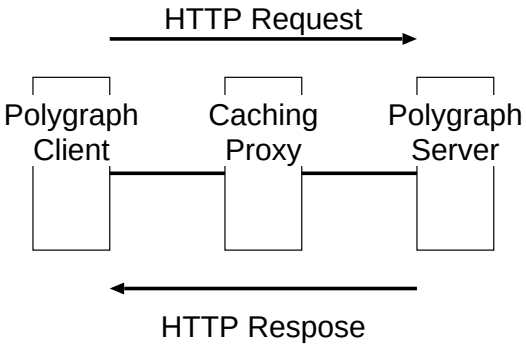

図 2: polygraph の測定方法

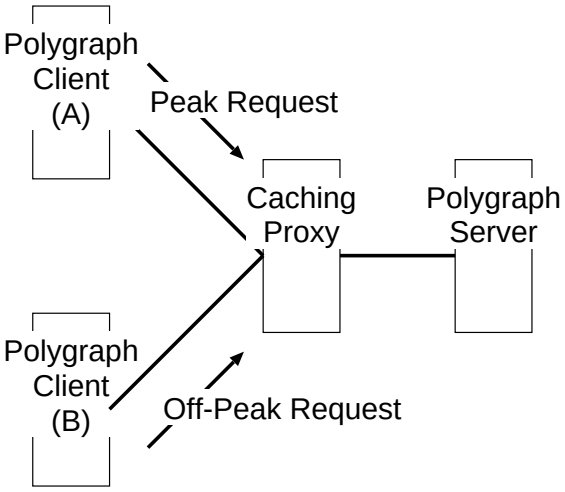

図 3: ピーク生成用のマシン構成

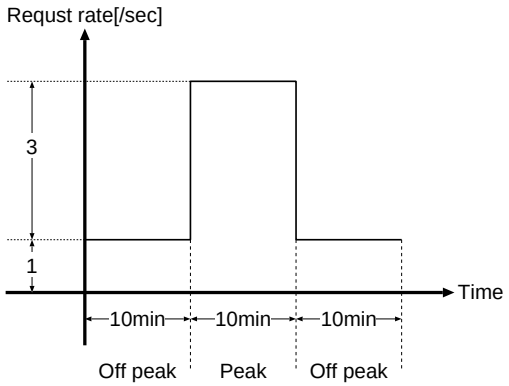

図 4: 想定したアクセスモデル

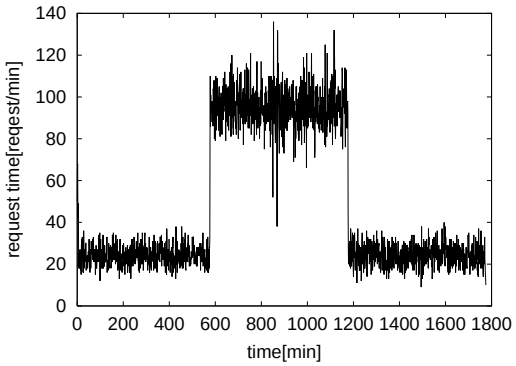

図 5: 実際に生成したアクセスパターンの例

間に代理サーバを挟んで測定を行う。

本実験では 2 つのクライアントを使用した (図 3)。一方のクライアントではオフピークのリクエスト生成を、もう一方のクライアントではピークのリクエスト生成を行った。

クライアントによるリクエストの生成方法を説明する。オフピークのリクエストを生成するためのクライアントを最初に起動する。オフピークを生成するクライアントは、30 分後に停止するようにしておく。オフピークを生成するクライアントが起動してから 10 分後に、ピークを生成するためのクライアントを起動する。ピークを生成するクライアントでは、オフピークを生成するクライアントの 3 倍のリクエストレートを生成させ、10 分後に停止するようにしておく。

ここで、リクエストレートとは、一秒間に発生するリクエストの数である。polygraph は、リクエストの生成速度の指定にリクエストレートを用いていることから、アクセスの度合を表す尺度としてリクエストレートを採用した。

以上の方法により、ピークを挟んで前後に 10 分間のオフピークの時間があるアクセスパターンを生成する (図 4)。

図 5 は、実際に生成されたアクセスパターンのリクエストレートを測定したものである。図 4 にかなり近い形状が得られている。従って、この方法により想定したモデルに近いアクセスパターンの生成が可能である。

3.2 応答時間の計測

本実験では、代理サーバの性能の尺度として応答時間を採用する。しかし、polygraph は個々の HTTP リクエストの応答時間を計測することはできないため、応答時間の計測には ENMA [7] を利用した。ENMA とはパケットモニタリングにより WWW サーバの性能評価を行うツールである。ENMA を用いることにより、個々の HTTP リクエストの応答時間を測定することが可能となる。

代理サーバを介したとき、キャッシュにヒットしなかった場合には HTTP の通信の流れは図 6 のようになり、代理サーバとオリジンサーバ間の通信が発生する。次に、キャッシュにヒットした場合、図 6 の (b) のパケット（リクエストの最後）を受信したら、以降のオリジンサーバとの通信を省略して、図 6 の (c) のパケット（最初のデータ）を返すことになる (図 7)。

本論文では、代理サーバがクライアントから最初の SYN（図 6 および 7 の (a)）を受けとってから、代理サーバがレスポンスの最初のパケットを送信するまで（図 6 および 7 の (c)）の時間を代理サーバの応答時間と定義する。

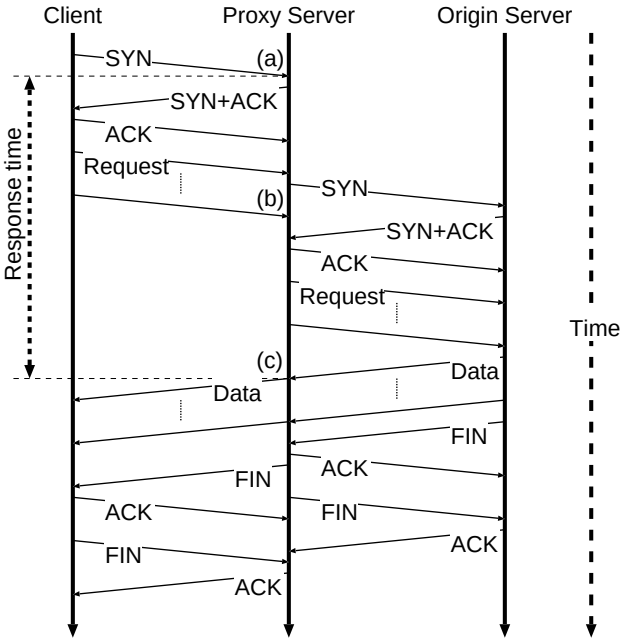

図 6: キャッシュミスしたときの HTTP の通信

3.3 測定方法

図 8 は、本実験に用いたマシン構成である。

3.1 節で述べたように、2 つのクライアントを用い、それぞれ別のホストを用意した。クライアントのリクエストレート、起動時間以外のパラメータとして、ヒット率は 55%、キャッシュの可能率[1]は 80%、リクエストの生成モデルとして zipf を指定した。

サーバは、クライアントとは別のセグメントに接続した。サーバのレスポンスを数秒間遅らせることにより WAN による遅延をエミュレートしている。本実験では、サーバはリクエストを受けとってから、ランダムに 1.5〜4.5 秒の間隔を空けてオブジェクトのデータを返す。

なお、ヒット率、キャッシュ利用率、リクエス

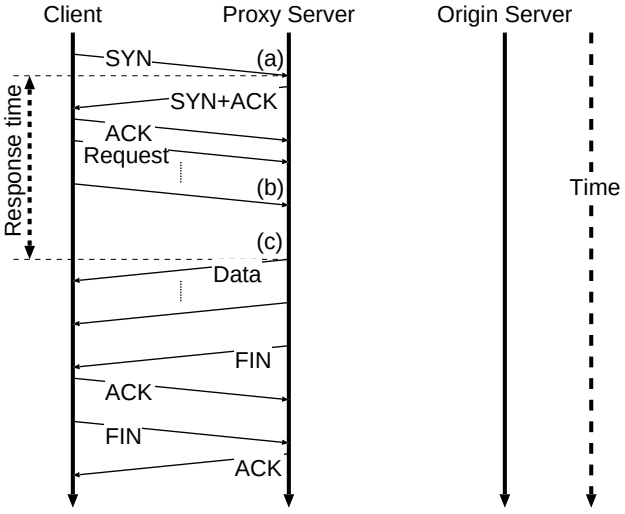

図 7: キャッシュヒットしたときの HTTP の通信

[1]リクエストヘッダに Progma: no-cache や Cache-Control: no-cache が指定されていない割合

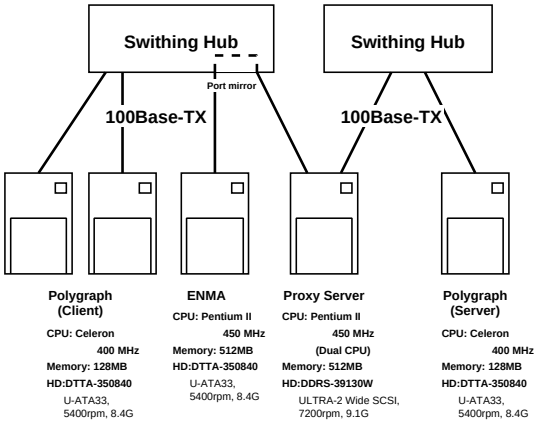

図 8: 測定マシン環境

表 1: 使用した主なソフトウェアとそのバージョン

OS	FreeBSD	3.2-RELEASE
代理サーバ	Squid	2.2 STABLE4
ベンチマーク	polygraph	1.3.1
応答時間計測	ENMA	19990823

ト生成モデル、サーバの応答の遅延時間の各パラメータは、First Bake-off [3] を参考にした。

代理サーバの応答時間の計測に用いるマシンは、クライアント側セグメントのスイッチングハブに接続されており、クライアントと代理サーバ間で流れるパケットをモニタリングできるようになっている。これは、スイッチングハブのポートミラー機能を利用して実現されている。

測定に用いたソフトウェアとそのバージョンを表 1 に示す。本実験では、キャッシング代理サーバのソフトウェアとして、一般に広く利用されている Squid [8] を利用した。Squid は、メモリとディスクを併用して WWW オブジェクトのキャッシュを行う。本実験では、メモリベースのキャッシング代理サーバの測定結果との比較のため、ディスクでのキャッシュ領域を 100Mbyte、メモリではキャッシュしないという設定で測定を行った。

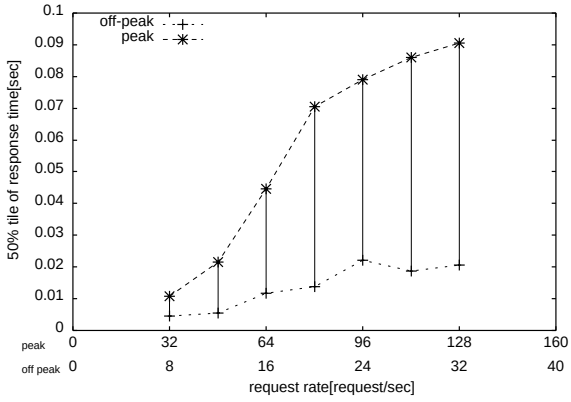

図 9: Squid(ディスクのみ) の応答時間の測定結果

3.4 結果

図 9 のグラフは、Squid の応答時間を測定した結果である。横軸はピークのリクエストレートを表している。この値の 1/4 が、オフピークのリクエストレートとなる。縦軸は応答時間であり、ピークとオフピークそれぞれの応答時間の 50% tile を示す。グラフ中の各測定点にある線分は、ピークとオフピークの応答時間の差を表している。

リクエストレートが増加するに従って、急激にピークの応答時間が悪くなっている。また、ピークとオフピークの差も急激に上昇している。

この測定では、130[request/sec] 以上のリクエストレートで測定を行うと、タイムアウトによるクライアントからのコネクション切断が大量に発生した。そのため、これ以上のリクエストレートでの測定は、測定結果から除外した。

またこのことから、130[request/sec] 以上のリクエストが発生すると、全てのリクエストを処理しきれなくなるので、代理サーバの応答が著しく劣化することも分かった。

4 ピーク時における応答時間劣化の要因

ピーク時における応答時間劣化の要因として、ディスクキャッシュによるディスクアクセスが考えられる。アクセスが少ないときは、同時に処理

するセッションの数が少ないので、ディスクI/O待ちが発生しても、大きな影響を受けることはない。ところがアクセスが増加してくると、ディスクI/O待ちのセッション数が増加してくるため、タイムアウトによるコネクションの切断が頻繁に発生する。そのため、代理サーバからの応答が悪化すると考えられる。

この問題を回避する手段として、従来から利用されているキャッシング代理サーバソフトウェアの中には、メモリとディスクを併用してキャッシュを行うものがある。この手法は、アクセス頻度の高いキャッシュをメモリ内に保持することにより、応答時間の高速化を図っている。この手法は、アクセスの少ない時は応答時間の高速化が可能である。しかし、アクセスレートが上昇すると、ディスクI/Oの影響が大きくなるため、ピークではメモリによる高速化の効果を失ってしまう。従って、メモリとディスクの併用では、ピークの応答時間劣化の改善は困難である。

5 メモリベース・キャッシング

メモリ領域のみを使ってWWWオブジェクトのキャッシングをすることにより、4章であげたディスクアクセスによる応答時間劣化の解決が可能である。

すでに、メモリベースのキャッシングについての提案はあるが、実装された例はない。そこで、実際にメモリベース・キャッシングを行う代理サーバを実装し、3章の方法で測定を行う。測定で得られた結果を3章の結果と比較しメモリベース・キャッシングの有効性を示す。

5.1 ピーク時の応答時間の劣化の対策

4章でピークの応答時間の劣化の原因として、ディスクアクセスを挙げた。ディスクアクセスによる応答時間の劣化を防ぐ方法として、メモリだけを用いてキャッシュを行う手法が考えられる。

メモリへのアクセスは、ディスクと比較して非常に高速である。従って、メモリベース・キャッシングでは、ピークの応答時間の劣化を緩和するだけではなく、平常時の応答時間も向上すると考えられる。

5.2 設計と実装

メモリベース・キャッシングにより、ピークの応答時間の劣化を緩和できるかどうかを確認するために、メモリベース・キャッシング代理サーバの実装を行った。メモリベース・キャッシングを実装するにあたり、既存の代理サーバの改変による実装ではなく、新たに代理サーバを構築した。

実装した代理サーバの特徴を挙げる。本実装では、selectシステムコールによる非同期I/O多重により、シングルプロセスで複数セッションの処理を行う。この手法は、Squid等でも採用されている。キャッシュサイズは固定長で、キャッシュエントリ数も固定である。キャッシュ領域は、起動時にヒープ領域から獲得される。キャッシュエントリが全て使われている状態で、キャッシュエントリに追加があると、キャッシュの置換が発生する。キャッシュの置換アルゴリズムは、LRUとした。

また本実装は、FreeBSD上で開発を行っているが、OS固有の機能は使用していないので、他のOSへの移植は容易である。

5.3 測定手順

我々は、メモリベース・キャッシングがピーク時の応答時間劣化に対して有効であることを示すために、実装したキャッシング代理サーバに対して、3章と同様の測定を行った。

3章の結果との比較のため、キャッシュサイズが100MByteになるように一つのオブジェクトのためのキャッシュサイズとキャッシュエントリ数を調整した。一つのオブジェクトのためのキャッシュサイズを15KByteとし、キャッシュエントリ数を6826個とした。その結果、全体のキャッシュサイズは、$15[\text{KByte}] \times 6826[\text{個}] = 102390[\text{KByte}]$となる。

本実装ではOSの仮想記憶上にキャッシュが構築されている。従って、純粋に物理メモリのみを使用しているわけではなく、スワップによるディスクアクセスが発生する可能性もある。しかし、3章の図8にあるように、代理サーバに使用したマシンには512MByteの物理メモリを搭載しているため、スワップの発生はないものとする。

5.4 結果

図10のグラフは、新たに実装したメモリベースキャッシング代理サーバと先に行ったSquidの応答時間の測定結果(図9)を同じグラフ上に表示したものである。横軸縦軸、および点の描き方は図9と同様である。

128[request/sec]まで測定しているグラフ(図10のSquid)がSquidの応答時間であり、288[request/sec]まで測定しているグラフ(図10のour system)が実装したメモリベース・キャッシング代理サーバである。

グラフの結果から、メモリベース・キャッシングによりピークにおける応答時間劣化が緩和されていることが分かる。また、オフピークの応答時間も大幅に向上していることも分かる。

以上の結果から、メモリベース・キャッシングはピークにおける応答時間の劣化を緩和できることが明らかになった。また、オフピークの応答時間を向上することが可能であることも明らかとなった。

5.5 実装の差による影響の検討

記憶装置以外の実装の違いによる影響を調べるために、Squidの記憶領域をFreeBSDのMFS(Memory File System)上に設計した場合の測定を行った。これによって、Squidでメモリベース・キャッシングのエミュレートが可能である。ただし、Squidはディスク上にキャッシュを構築するようになっているので、この設定はかなり特殊であり、運用方法としては多少公平さに欠けるところがある。

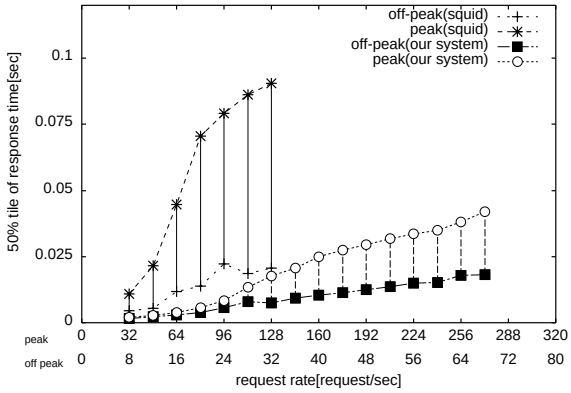

図10: メモリベース・キャッシング代理サーバの応答時間の測定結果

測定結果を図11に示す。測定は、3章と同様の方法で行った。縦軸、横軸等は、図9、10と同じである。グラフから、Squidでもメモリを使ってキャッシュすることにより、応答時間劣化を緩和できることが分かる。従って、実装とは独立に、用いる記憶装置によって、ピーク時の応答時間劣化が緩和されることが分かる。

6 今後の課題

メモリベース・キャッシングにより、ピークの応答時間の劣化を防ぐことが可能であることが分かった。しかし、いくつかの課題が残っている。

代理サーバとWWWサーバ間のWAN環境のエミュレート方法が挙げられる。本実験では、代理サーバとWWWサーバ間を100Base-TXのLAN環境にして実験を行っている。そのため、WAN環境をエミュレートするために、サーバは要求の受信後、数秒間待ってから受信した要求に対する応答を返すように設定した。しかし、これだけではWAN環境のエミュレートは、不十分であり、狭い帯域幅や劣悪な通信路をも考える必要がある。Second Bake-off [4]では、このようなWAN環境のエミュレートにDummynet [5]の採用を予定している。そこで、我々もDummynetのようなツールを用いて、厳密にエミュレートしたWAN環境上で実験を行う必要がある。

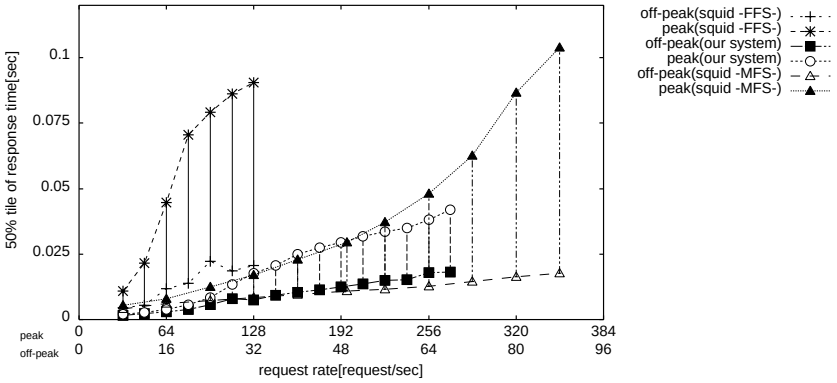

図 11: MFS 上にキャッシュする Squid との比較

また、測定上の課題として、オフピークからピーク、ピークからオフピークに移行する時の過渡状態に対する評価を行なう必要もある。今回用いた polygraph では、リクエストレートの細かい制御ができなかったため、凸型のピークしか生成できなかった。しかし、最近リリースされた polygraph の新しいバージョン (2.x) では、リクエストレートのより細かい制御が可能となった。従って、凸型ではなく、なだらかなリクエストレートの上昇、下降モデルの生成が可能となり、これより過渡状態の測定が可能であると考えられる。

最後に、実装上の課題として、キャッシュ容量の問題がある。キャッシュに必要なディスク容量は、100 人規模のサイトで 1〜1.5GBytes、3000 人規模のサイトでは 3〜6GBytes は必要であるといわれている [10]。ディスクであれば、余裕のある数値であるが、ディスクの 1/10 倍から 1/100 倍程度の容量であるメモリの利用を考えると現実的に厳しい数値である。従って、小さいキャッシュ領域をできる限り有効に活用できるアルゴリズムを開発する必要がある。これには、オフピークの時は Squid 等と同様にメモリとディスクの併用を行い、ピークになるとメモリベースキャッシングに切替えるという手法が考えられる。しかし、この方式には、ピークとオフピークの判定方法が極めて重要である。

7 まとめ

代理サーバのアクセスには時間的に偏りがあり、一日に何度かアクセスの集中するピークが存在する。そのピークになると代理サーバの応答が悪化する。

まず、我々はピークとオフピークで応答時間にどの程度の差があるのかを測定した。測定には、ベンチマークソフトで作った凸型のアクセスモデルを利用した。ピークのアクセスレートをオフピークの 4 倍にして測定を行ったところ、オフピークのアクセスレートの上昇とともにピークとオフピークの応答時間の差は広がっていくことが明らかとなった。

次にピークの応答時間劣化の原因となるディスクアクセスに注目し、その対策としてメモリベース・キャッシングを用いることを提案した。そして、メモリベース・キャッシングの有効性を示すために、メモリベース・キャッシング代理サーバを実装し、応答時間の測定を行った。その結果、メモリベース・キャッシングにより、ピークの応答時間劣化を抑えることが可能であることを明らかにした。また、オフピークの応答時間が向上できることが明らかとなった。

謝辞

本研究に際して、快く実験機材の提供をして頂き、また数々の有意義な助言を頂きました日立製作所の吉田様、西門様、田口様、安齋様に感謝致します。

また、数々の有意義な助言を頂きましたWIDEプロジェクトの3WAワーキンググループの方々に感謝致します。

最後に、研究を進めていくうえで貴重な助言を頂きました奈良先端科学技術大学院大学の山口英先生に感謝致します。

参考文献

[1] N. Nishikawa, T. Hosokawa, Y. Mori, K. Yoshida, H. Tsuji,

"Memory-based architecture for distributed WWW cachig proxy",

7th International World Wide Web Cunference, 14-18 April 1998, Brisbone

[2] B. L. Wong,

"Sizing up your Web server",

SunWorld, October 1997,

http://www.sunworld.com/sunworldonline/swol-10-1997/swol-10-sizeserver.html

[3] A. Rousskov, D. Wessels, G. Chisolm,

"The First IRCache Web Cache Bake-off",

April 1999,

http://bakeoff.ircache.net/

[4] "The Second Bake-off",

January 17th 1999,

http://bakeoff.ircache.net/N02

[5] Luigi Rizzo

"Dummynet a simple approach to the evaluation of network protocols",

ACM Computer Communication Review,Vol.27,n.1,January 1997,pp.31-41

[6] IRCache,

"Web Polygraph",

http://polygraph.ircache.net/

[7] Yutaka Nakamura, Ken-ichi Chinen, Hideki Sunahara, Suguru Yamaguchi, Yuji Oie

"ENMA: The WWW Server Performance Measurement System via Packet Monitoring",

INET'99 Internet Society, 22-25 June 1999, SanJose, Calfornia, USA

[8] "Squid Web Proxy Cache",

http://www.squid-cache.org/

[9] M.Abrams, C.R.Standridge, G. Abdulla, S Williams, and E.A.Fox

"Caching Proxies: Limitations and Potentials",

Technical Report, Virginia Polytechnic Inst. and State University, TR-92-12,1995

[10] A. Luotonen, 前田 奈美 訳, 渡辺 知夫 監修,

「Web プロキシサーバ パフォーマンスの最適化とセキュリティ」,

プレンティスホール出版,1998

[11] W. R. Stevens

"UNIX Network Programing Volume 1 second edition",

Prentice Hall ,1997

[12] R. Fielding, H. Frystyk, T. Berners-Lee

"Hypertext Transfer Protocol – HTTP/1.0",

RFC1945, May 1996

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee,

"Hypertext Transfer Protocol – HTTP/1.1",
RFC2068, August 1997