2004－DPS－116　(7)
2004／1／29

複数のミラーサーバへの効率的なアクセス機構の実現

藤澤 弘† 最所 圭三‡

香川大学大学院工学研究科† 香川大学工学部‡

E-mail: {s02g475@stmail.eng.kagawa-u.ac.jp, sai@eng.kagawa-u.ac.jp}

パーソナルコンピュータの急激な普及およびインターネットの急激な拡がりは多くの Web ページを登場させ、特にアクセスが多いサイトでは同じサービスを行うミラーサーバを用意している。本稿においては、これらのミラーサーバを有効活用し、信頼性の高い接続および柔軟な接続を実現するプロキシ機構を提案し、特に多くの画像などが表示される Web ページを表示するための効率の向上を図りその評価を行った。その結果、提案した機構の有効性を確認できた。
キーワード: プロキシ、高効率、持続的接続、HTTP パイプライン

Implementation of Efficient Access Mechanism for Multiple Mirror-Servers

Hiroshi FUJISAWA†and Keizo SAISHO‡

†Graduate School of Engineering, Kagawa University ‡Faculity of Engineering, Kagawa University

E-mail: {s02g475@stmail.eng.kagawa-u.ac.jp, sai@eng.kagawa-u.ac.jp}

Rapid diffusion of personal computers and expanding the Internet introduced many Web servers. Many popular sites accessed by a lot of clients prepare mirror servers that provide same services. In this paper, a proxy mechanism which supports reliable and flexible connections is proposed in order to realize load balancing on mirror servers. Especially,a method to reduse time to open Web pages with many images is described. The results of experiments show effectiveness of proposed proxy mechanism.
Keywords: Proxy, Highly reliable, Persistent Connection, HTTP pipelinig

1　はじめに

現代の情報化社会におけるパーソナルコンピュータの急激な普及とコンピュータ通信であるインターネットの拡がりは加速の一途をたどっている。それに従い、多くの個人や企業が Web ページを開設し、広く役立てている。さらに、近年のインターネット人口の増大、ADSL をはじめとするブロードバンドの急激な普及によるネットワークトラフィックの負荷が増大により、Nimda や CodeRed などのマイクロソフト社製 Web サーバを狙ったワームによる被害や、DoS 攻撃のような外的要因によりサーバがダウンしたり、メンテナンス等の内的要因によりサービスが提供できない状況が考えられる。

これらの問題のために、新聞社や検索エンジンのサーバーや、大手企業の人気サーバーなどでは、ミラーサーバを用いている。ミラーサーバとは、同じ情報を提供するようにデータをコピーした (ミラーリングした) Web サーバを複数用意し、アクセスを分散することによって負荷分散、信頼性の向上を行っているサーバのことである。

ここでいう信頼性とは、ミラーサーバの一部が障害を起こしても、通信できることを確保することである。しかし、実際にはミラーリングされたサーバが存在しても、実際にアクセスする場合はそのうちの１つにアクセスすることになる。このため、接続したサーバによって応答が悪くなったり、最悪の場合停止することもある。このように、サーバーがミラーリングされていても負荷分散、信頼性向上に関して有効に活用されていないことが考えられる。

この問題を改善するため、本研究ではミラーリングされた複数のサーバに同時にアク セスすることにより、ミラー サーバの負荷分散及びアクセスの信頼性を

向上させる方法について提案している [1, 2]。本稿では、提案した方法を実現するためのプロキシ機構の設計および実現について報告し Web サーバからのファイル取得の時間短縮、効率の向上について、インターネット上のミラーサーバ及び大学研究室内のミラーサーバで評価を行う。

2　プロキシ機構の概要

提案するプロキシ機構 (以下、プロキシ機構) は、通常の Web サーバに対する要求の場合は、その要求をそのまま Web サーバに転送し、Web サーバからの応答をそのままクライアントプログラムに転送する。ミラーリングされた Web サーバに対する要求は、アクセスするデータの性質、サーバやネットワークの状態によりサーバを選択したり、アクセスの方法を変更したりする。このような機能を持ったプロキシ機構を用いることにより、どのような HTTP クライアント (Web ブラウザなど) に対しても、透過的に Web サーバにアクセスすることができる。

以下、プロキシ機構で実現する Web サーバへのアクセスの方法について述べる。

2.1　高信頼アクセス機構

一部のミラーサーバが停止してもクライアントに気づかせない機構である。プロキシ機構はミラーリングされているサイトにアクセスする場合、ミラーサーバのアドレスを保持している。対象のミラーサーバが停止していることを検出すると、他のサーバに自動的に切り替える (図 1)。プロキシ機構では、通信エラーが発生した場合にサーバが停止したとみなす。

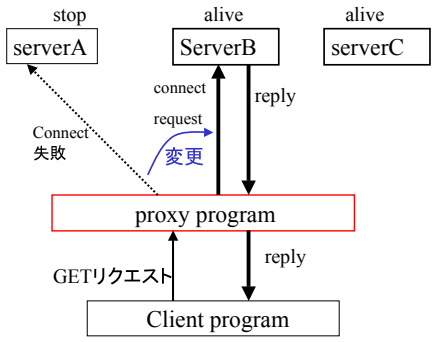

図 1: 通信エラーによるサーバの自動切り替え

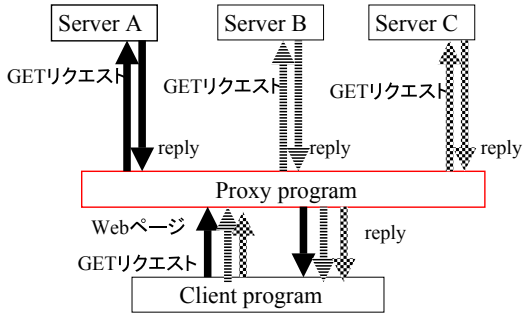

図 2: 振り分け接続

2.2　柔軟なアクセス機構

複数のミラーサーバを有効に利用するための機構である。以下に示すようなアクセス方法を提供する。

選択接続

ミラーサーバに対して、定期的に応答時間を調べておき、最も応答時間が短いサーバに対して要求を送ることにより応答時間の短縮および負荷分散を行う。

振り分け接続

Web ページの多くは、テキストデータである HTML ファイルだけでなく画像や音声、動画などの複数のコンテンツの集合として構成されている。現在のクライアントの多くは、それらのコンテンツを同時にサーバに要求する。そこで、図 2 に示すように、連続した要求を、複数のミラーサーバに振り分けて同時に要求することで、Web ページ全体の取得時間を短縮する。

分割取得

1 つの巨大なファイルを分割し、それらを異なるサーバに対して同時に要求し、応答結果をプロキシ機構内で再構築したあとクライアントに返す。これにより、ファイルの取得にかかる時間を短縮する。Windows 上のフリーソフトである分割ダウンローダ (Irvine など) と同等の機能である。

本稿では、上記で示したアクセス機構を持つプロキシ機構を設計し実現する。

3　プロキシ機構の設計

本稿では、2 節で示したアクセス機構のうち最も利用頻度が高いと思われる振り分け接続の設計について述べる。

Web アクセスの基本プロトコルである HTTP 1.0 では、HTML のソースコードや画像ファイル、音声ファイルなど、各オブジェクトの送受信ごとに個別に TCP コネクションをオープンしていたので負荷が大きい TCP のオープンやクローズ処理を何度もしなければならずそのためのオーバーヘッドが大きなものである。そのような問題を解消するため HTTP 1.1 では、以下の拡張がなされている。プロキシ機構はこれらの機能に対応する。

3.1 HTTP の追加機構

HTTP 1.1 で追加された機能の主なものは以下の 2 である。

1. 持続的接続 Persistent Connection
   HTTP 1.0 では、Web サーバに対する 1 回の応答が終了した時点で即座にそのサーバに対する接続は切断される。そのため、何度も同じサーバに要求を行う場合は、接続と切断を繰り返すため、そのオーバヘッドにより、通信効率が低下する。

   HTTP 1.1 では、RFC2616 に規定されている持続的接続の機能が追加された。この機能により、従来では HTML のソースコードや画像ファイル、音声ファイルなど、各オブジェクトごとに個別に TCP コネクションをオープンしていたが、これを 1 つの TCP コネクションで連続して取り扱うことができる。この機能により、1 回の応答が終了した後でも接続を維持している。TCP 接続がすでに確立しているので、次回のリクエスト時には TCP コネクションのオープンの時間が必要で無く、リクエストへの反応時間は短縮できる。同じサーバに対して連続した要求が送られた場合、従来に比較して高速なデータ転送が可能である。

2. HTTP パイプライン
   複数の HTTP リクエストを、対応するレスポンスを待つことなくソケットに同時に書き出すことを許可する HTTP1.1 の機能である。この機能を使うことにより、1 つの HTTP リクエストのレスポンスを待つことなく、図 3 のように、同時に複数のリクエストを送信することができる。

以上の機能により、Web アクセスの効率が向上する。

```
GET /google/lang.files/kr_flag.gif HTTP/1.1
Host: saisyolab

GET /google/lang.files/co_flag.gif HTTP/1.1
Host: saisyolab

GET /google/lang.files/jp_flag.gif HTTP/1.1
Host: www.google.co.jp
```

図 3: HTTP パイプライン要求の例

3.2 通信プロトコルの設計

本節では、以上の機能を追加したプロキシプログラムでの通信手順を設計する。持続的接続および HTTP パイプラインの要求が来た場合の動作の概念図を、図 4 と図 5 に示す。

まず図 4 で示される持続的接続を用いた振り分け接続について説明する。当初は、当初は、本接続のみに対応したプロキシ機構を実現した [1]。図 4 では、まず①で複数の要求がクライアントからプロキシへ来る。プロキシ機構は、複数の要求を振り分けてそれぞれの Web サーバへ接続し、送信する (②)。③で、それぞれのサーバから受け取った応答をプロキシがクライアントに渡す。その後、連続して要求が来た場合には、新たに Web サーバに接続する必要無くリクエストを送信することができ、そのレスポンスを受け取りクライアントへ送信するという動作を繰り返す (④)。そして、すべてのデータのやりとりが終った後で、サーバとの接続の切断を行う (⑤)。

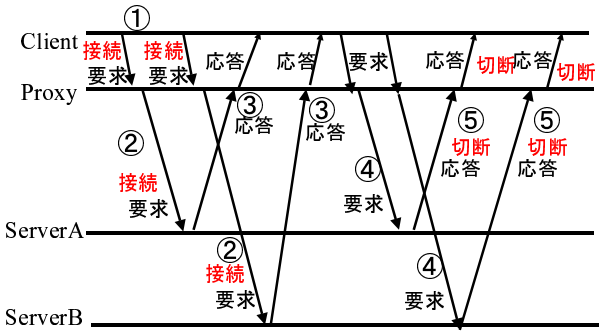

図 4: 持続的接続を用いた振り分け接続

しかし、この方法のプロトタイプを実装し、測定した結果では、あまりファイル取得の時間短縮の効果は見られなかった。そこで、HTTP パイプラインの機能を取り込むことにした [2]。それが、図 5 で示されるも

のである。最初の要求 (①) でページを読みだし、クライアントに送る。この後、プロキシ機構は次の HTTP パイプラインを用いるアクセスに備え別のミラーサーバとの TCP コネクションをオープンしておく。クライアントはそのページを表示するために複数のデータが必要なことを判断し、プロキシに HTTP パイプラインで要求する (②)。プロキシは、HTTP パイプラインの要求を分解し、複数のミラーサーバに送る (③)。このとき、最初に接続していたミラーサーバ以外は、接続した後要求を送る (③', ③")。その後、サーバから帰って来たレスポンスを HTTP パイプラインリクエストの順番通りにクライアントに渡す (④)。

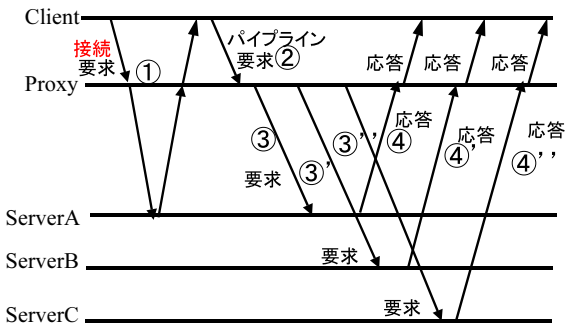

図 5: HTTP パイプラインを加えた振り分け接続

4 実装

前章で設計した振り分け接続の実装について述べる。HTTP 1.0 で接続要求が来た場合は要求毎にコネクションをオープンする。

HTTP 1.1 で接続要求が来た場合は、最初の要求の時に TCP コネクションをオープンし、それを維持する。これにより持続的接続を実現する。プロキシプログラムは、サーバへ接続するときにサーバ情報を格納する構造体に接続したファイルディスクリプタと接続ときの時刻をアクセス時刻として保持しておく。続けてリクエストがクライアントから来た場合は、そのファイルディスクリプタを使いリクエストを行い、アクセス時刻を更新する。アクセス時刻が、設定した時間を越える、もしくは、Web サーバからのレスポンスに持続的接続を切断するというオプションがあれば、そのサーバとの接続を切断しサーバ情報を格納する構造体を更新する、新たなリクエストが来た場合は、新たにサーバへ接続する。

クライアントから複数の接続が来た場合は、スレッドを生成しそれらに処理を任せる。各スレッドは、異なるミラーサーバに接続する。これにより、振り分け接続を実現する。

次に HTTP パイプラインの処理について述べる。本機能はスレッドを用いて実装した。最初の HTTP 要求 (図 5 の①) は、メインのスレッドにより処理する。このとき、オープンした TCP コネクションは維持する。そして、メインスレッドでファイル要求、取得の処理を行っている間に、mozilla のデフォルトの HTTP パイプライン要求の値 4 − 1 個 = 3 個のスレッドを生成し、それらのスレッドでミラーサーバに TCP コネクションをオープンしておく。このとき、ミラーサーバの個数が HTTP パイプライン中の要求よりも少ない場合は、複数のスレッドが同じミラーサーバに TCP コネクションを行う。HTTP パイプライン要求が来た場合、メインのスレッドと先に生成しておいたスレッドを用い要求を処理する。次の HTTP パイプラインが来たら、そこに含まれる HTTP 要求が処理をしているスレッド数より多い場合は、新たなスレッドを生成する。逆に少ない場合は、要求を割り当てないスレッドが生じるが、そのスレッドはそのまま接続を維持し、次の要求に備える。

これらを、Linux 上で C 言語を用いて開発した。ソースレベルで 1700 行程度になった。

5 関連研究

本稿で述べるシステムは、Web のミラーサーバに効率的にアクセスするためのローカルプロキシプログラムである。それらに関連する研究には以下のものがある。

- みかん

  みかん [3] とは、広島市立大学で開発された最新ファイルを近傍のミラーサーバから取得するために使用する仮想的な FTP サーバである。 FTP クライアントから接続要求が発生した場合、みかんが自動的に近傍のミラーサーバの撰択を行い、FTP クライアントの代わりにファイルの取得を行い、それを FTP クライアントに返す。FTP クライアントには、みかんから FTP サービスを提供されているようにしか見えない。 従来の FTP クライアントから使用可能であるという利点がある。本稿で研究しているプロキシ機構は HTTP を対象としており、みかんとは対象が異なっている。

- Load Balance

  DNS の機能として、DNS サーバが一つのドメイン名に複数の IP アドレスを割り当てて、問い合わせがあるたびに順番に答えることにより、1 台のコンピュータへアクセスが集中するのを防ぐラウンドロビン DNS がある。１つのサーバ名に対して、同じサービスを行っているミラーサーバを登録することにより負荷分散が実現できる。しかし、それだけでは負荷分散は十分なものではなく、各社でロードバランサ（負荷分散装置) が開発されている。DYNAMIC LOAD BALANCING ON WEB-SERVER SYSEMS [4] などの効率的な負荷分散方法が研究されている。これらの機能は、コネクション毎に接続を振り分けることができ負荷分散の効果があるが、本稿で述べている機能ではさらに HTTP の要求毎に接続を振り分けることができる。

- DNS フィルタ方式によるミラーサーバ選択法

  DNS フィルタ方式 [5] とは、筑波大学で研究されているミラーサーバ選択法のことである。このシステムは、ローカル DNS キャッシュサーバにフィルタ機能を持たせ，クライアントからの DNS 問合せに対する上位 DNS サーバの応答から，ユーザにとって適切なサーバを自動的に取捨選択する DNS フィルタ方式を用いてる。そして，クライアントから見て最も短い時間でファイルを転送するサーバを最適とするシステムである。しかし、DNS キャッシュサーバを用いているので、ただ一つの最適なサーバを選択するためには適しているが、一つのクライアントが複数のサーバを同時に利用し、協調動作させてさらに通信効率を上げるということは難しい。

6 評価

開発したプロキシプログラムに時間測定の機能を加え、それぞれの通信動作にかかる時間を測定した。評価にあたり、実際のインターネット環境で運営されているミラーサーバであるグーグル (www.google.co.jp) と、香川大学最所研究室内でミラーサーバを用意した。表示するためのページとしては、両方とも HTML ファイルと８０個の画像を開くページ (合計 88.9kBytes) を用意した。

以下の環境で評価を行った。

対象 Web サーバ

グーグル	香川大学 最所研究室
GWS 2.1	Apache
216.239.39.99	133.92.144.40
216.239.57.99	133.92.144.37
216.239.33.99	133.92.144.49
216.239.53.99	133.92.144.58

評価用クライアントマシン

	グーグル	香川大学 最所研究室
	ADSL 12Mbps 実効 3 Mbps	LAN 100 Base-T
CPU	Duron 1.1GHz	Pentium III 550MHz
メモリ	256MByte	メモリ 256MBytes
ネットワーク	ADSL 3MBps	100 Base-T LAN
ブラウザ	Mozilla 1.4.1	Mozilla 1.5

以上の環境で、以下の１から５の実験を行った。

1. 全ての機能を用いない接続方法
2. 持続的接続のみを用いる方法
3. 持続的接続と振り分け接続を利用した方法 (図 4 で示した方法)
4. 持続的接続と HTTP パイプラインを利用した方法
5. 持続的接続、HTTP パイプラインおよび振り分け接続の機能を利用した方法 (図 5 で示した方法)

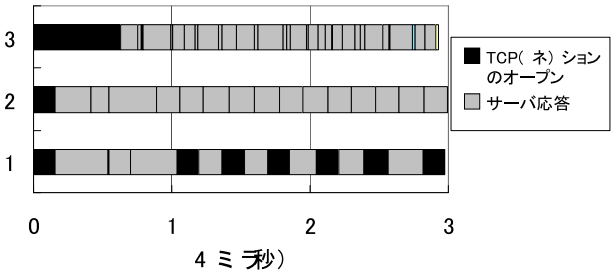

図 6: 実験 1、2、3 の比較

グーグルとのアクセス結果について評価する。図 6 は、実験 1、実験 2 および実験 3 においてサーバへのアクセスを開始してから最初の 3 秒間のサーバへの接続とデータ転送の様子を示したものである。黒い部分がサーバとの接続に要する時間であり、グレーの部分

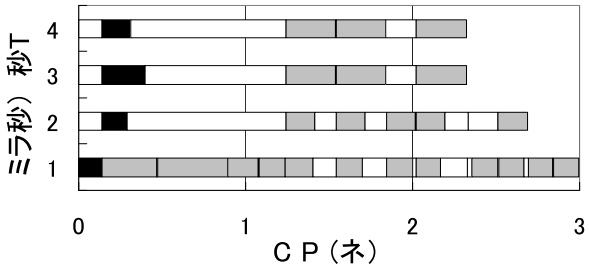

図 7: HTTP パイプラインを使った振り分け接続 (実験 5)

がデータ転送に要する時間である。この中では、振り分け接続を用いた持続的接続が効果的であることが確認できる。図 7 は、持続的接続、HTTP パイプラインおよび振り分け接続の全ての機能を用いた方法のサーバへのアクセスを開始してから最初の各ミラーサーバへのサーバ接続、データの応答による時間の最初の 3 秒間を示している。この結果より、ブラウザからの複数のリクエストをそれぞれのサーバに振り分けることができる振り分け接続が有効に活用できていることが分かる。しかし、2.4 秒過ぎてからブラウザからの要求で HTTP パイプライン機能が使われなくなってしまった。ただ、研究室内のサーバへのアクセスではこのようなことは起こらなかった。この原因については、調査中である。

さらに、これらの実験を 10 回程繰り返し、Web ページを開く時間を測定した結果を以下に示す。

実験 No	かかった時間	
	香川大学	グーグル
	100BaseT	ADSL 3Mbps
1	3.3	29.5
2	2.6	15.9
3	2.8	11.4
4	2.6	—
5	2.4	13.6

この中でグーグルにおける実験 4 の結果がないのは、グーグルの Web サーバである GWS が HTTP パイプラインの機能に対応していないからである。

香川大学の研究室内で行った実験では、持続的接続、HTTP パイプライン、振り分け接続の機能全てを使った方式が最も Web ページを開くのが早く効率が良いと言うことが確認できた。一方、インターネット上のグーグルを用いた実験では、持続的接続と振り分け接続の機能を使った方式が最も早いという結果となった。これは、先に述べた HTTP パイプラインの機能が途中で使用されなくなったためである。仮に HTTP パイプラインが最後まで続き 2 倍のデータ転送量が得られると仮定すれば約 4.6 秒 (=(13.6-2.4)/2) 短縮でき、約 9 秒で終了でき最も効率が良くなる。

このように、HTTP パイプラインや振り分け接続を使った接続方式にした場合で、通信時間を短縮できた。また、HTTP パイプライン要求に対応していないサーバであっても、提案したプロキシ機構を用いれば、HTTP パイプラインの機能を活用できることも確認できた。また、今回の実験では、ミラーサーバの数を 4 つとしたので、多くの画像を取得するページでは振り分け接続が有効となったと考えられる。

7　まとめ

複数のミラーサーバを有効活用し、信頼性の高い接続および柔軟な接続を実現するプロキシ機構を提案し、特に多くの画像などが表示される Web ページを表示するための効率の向上を行うための機能を考えた。それらの評価をインターネット及び LAN 上で評価を行った。その結果、提案した機構の有効性を確認できた。

今後の課題としては、今回の実験で判明した問題点の解決、そして、さらなる通信効率の向上、新たな通信方法の提案をしていきたいと考えている。

参考文献

[1] 藤澤弘、最所圭三. “持続的接続を用いた複数のミラーサーバへの効率的なアクセス機構について”, 平成 14 年度 電気関係学会四国支部連合大会論文集, 16-9, p.306, 2002.

[2] 藤澤弘、最所圭三. “複数のミラーサーバへの効率的なアクセス機構について”, 平成 15 年度 電気関係学会四国支部連合大会論文集, 16-24, p.325, 2003.

[3] 尾藤正人、舟阪淳一. “ みかん - ミラー選択機能付き代理 FTP サーバ”，Linux Conference 2002，2002.

[4] Valeria Cardellini, Michele Colajanni, Philip S. Yu, “Dynamic Load Balancing on Web-Server Systems”, IEEE Internet Computing, Vol.3, No.3,pp.28-39,1999.

[5] 横田 裕思、木村 成伴 、海老原 義彦. “DNS フィルタ方式によるミラーサーバ選択法の提案と実装”, インターネットコンファレンス 2001 論文集 p.121-130,1999.