WWW 分散キャッシュサーバ*

鳥谷 明雄† 西村 浩二‡ 相原 玲二‡

†広島大学 大学院 工学研究科 ‡広島大学 総合情報処理センター

WWW の利用は急速な成長を遂げている。その結果ネットワークバンド幅の過度の使用やサーバの過負荷を招くため、キャッシュサーバを導入し、トラフィックの削減、レスポンスタイムの減少に努めている。また、単独動作が基本のキャッシュサーバでは、キャッシュサーバの過負荷が問題になる。本論文では、複数のキャッシュサーバに負荷を分散させ、全体として1つの大きなキャッシュサーバを構成する方式を提案する。

1 はじめに

WWW (World Wide Web) はクライアントサーバモデルを基本とした情報検索システムであり、現在では指数的に急速な成長を遂げている。その結果、ネットワークの過度のバンド幅の利用、あるいはアクセスが集中することによる過負荷が生じている。この過度のネットワーク利用とサーバの過負荷のために、アクセス時の遅延が生じている。

これらのことを解決するためにキャッシュサーバの導入が一般的である。クライアントからの要求があるとキャッシュサーバはキャッシュが存在すればそのデータをクライアントに送信し、そうでなければそのオリジナルサーバからデータを取得、蓄積しさらにクライアントに送信する。キャッシュサイズが大きいほどキャッシュのヒット率は高くなる。しかしながら、1つのキャッシュサーバを使用すると、接続クライアント数に制限が生じ、キャッシュの効果が制限される。

以上のことから本論文では、複数のキャッシュサーバに負荷を分散させ、全体として1つの大きなキャッシュサーバを構成する方式を提案する。その際、キャッシュサーバをグループ化し、グループ間ではキャッシュの重複を許し、グループ内ではサーバ間で共通のインデックス情報を交換し、互いにキャッシュエリアがオーバラップしないよう協調動作する。一方、クライアントからのアクセスに対して、適当なサーバを選択し、リクエストを転送する機構を導入する。

*WWW Distributed Cache Server by Akio Toya†, Kouji Nishimura‡ and Reiji Aibara‡, †Graduate School of Engineering, Hiroshima University, ‡Information Processing Center, Hiroshima University.

2 背景

2.1 キャッシングプロキシ

インターネットに接続する場合、多く組織ではセキュリティを向上させるために防火壁 (firewall) を利用している。しかしながら、防火壁の構築により利便性が損なわれ、利用可能なサービスやユーザインターフェイスが制限される。そこで、ソフトウェアで実現した中継システムにより特定のアプリケーションのプロトコルを中継する方法が開発されて来た。その代表的なシステムとして HTTP (Hypertext Transfer Protocol) プロトコル中継機能つきの CERN-httpd[1]、いわゆるプロキシサーバが広く使われている。

また、アクセスのレスポンスタイムを減少するためキャッシュ機能が導入された。これは一度得たデータを一時的に保持しておき、同じデータを再度アクセスした場合それを再利用する。これにより遠隔サーバへのアクセスの無駄を省くことができ、ネットワークの負荷を軽減させる効果もある。そしてこのキャッシュ機能をもつ中継サーバはキャッシングプロキシ (図1) と呼ばれキャッシュサーバとして近年発達してきている。

2.2 従来のキャッシュサーバ

これまで開発されたキャッシュサーバの中で最も基本的なものは CERN (European Laboratory for Particle Physics) で開発された CERN-httpd である。この CERN-httpd はクライアントからのリクエストに対しキャッシュサーバにキャッシュがない場合には HTTP、FTP などのプロトコル

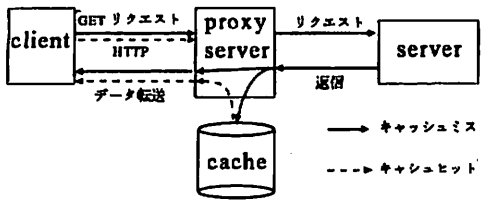

図 1: キャッシュ機能を持つプロキシサーバ

を用いてデータをサーバから受け取り、キャッシュサーバにデータをキャッシュし、クライアントにデータを送信する。キャッシュサーバにキャッシュがある場合にはキャッシュサーバのキャッシュデータをクライアントに送信する。

国内では、電子技術総合研究所で開発されたプロトコル中継システム DeleGate がある。このシステムは多くの中継システムで用意されている機能に加えて、NNTP、X、telnet のプロキシ、漢字コード変換の機能などがあって、多様な機能を備えていることが特徴である [2]。

LANR (National Laboratory for Applied Network Research) で開発された Squid と呼ばれるキャッシュサーバがある。これは、プロキシサーバ専用に開発されたため、高速なキャッシング機能とデータ提供機能を持っている。またネットワークに分散配置されたキャッシュサーバをキャッシュアクセス専用のプロトコル (ICP プロトコル) で接続し、一つのキャッシュシステムに見せる事ができる。さらにキャッシュ用ディスクを複数のディスクパーティションに分けることができる [3]。

この他にも、複数のキャッシュサーバを立ち上げておきクライアントがそれらに対してマルチキャストを行ない、キャッシュの存在するサーバを知りそこにリクエストを送信するもの [4]、キャッシュを shot-term キャッシュと long-term キャッシュに分け、さらにオリジナルサーバで内容が更新、削除されていればそれ知らせる機構を導入したもの [5] などがある。

2.3 問題点及び解決法

第 1 の問題点として、2.2のシステムでは、それぞれのキャッシュに TTL (time to live) つまり有効期限を設定しておき、その期限内ではキャッシュは新しいとみなすため、有効期限内にオリジナルのデータが更新されていてもキャッシュが更新されない可能性があることである。そこで、提案するシステムでは、ユーザに常に最新の情報を提供することによって、オリジナルのデータとキャッシュデータの一貫性を保つオプションを用意している。

次に、キャッシュサーバの一般的な利用ではクライアントがキャッシュサーバを予め指定しておくため、何らかの理由でキャッシュサーバがダウンした場合に、データが取得できないなどの問題が生じてくる。Squid では設定ファイルによりディスクをドメイン名毎に分割することが可能であるが、このようなディスクの分散を設定ファイルなしで行なえるようにした方が便利である。提案するシステムでは、近接のキャッシュサーバをグループ化しその中で複数のキャッシュサーバを協調させる。そしてクライアント側にキャッシュサーバの選択を行なう機構を導入することにより、システム全体としてキャッシュサーバの強靭さをはかると同時にディスクの分散を行なえるようにする。

さらに、それらのグループを複数持ち、キャッシュを交換する機構を導入することにより負荷の分散をはかる。

3 WWW 分散キャッシュサーバ

3.1 設計方針

次のような方針のもとで分散キャッシュサーバの設計を行なった。

- WWW のブラウザは商用化されたものも含め多数普及しているのでブラウザの改良は行なわない。
- LAN (Local Area Network) 内のキャシュサーバをグループ化し、それらグループ間ではキャッシュの重複を許す。
- グループ内では、同じ URL (Uniform Resource Locators) に対するキャッシュは 1 つしか持たない。

- クライアント側に適当なサーバを選択し、リクエストを転送する機構 (プロパゲーションサーバ) を導入する。
- キャッシュサーバは全て同じ機能をもつ。

3.2 システムの概要

提案するシステムの概略図を図 2に示す。このシステムでは、LAN 内で複数のキャッシュサーバをグループ化する。そして、そのグループを複数存在させる。また、1 つのクライアントに対し 1 つのプロパゲーションサーバを置く。

クライアントからのリクエストがあるとプロパゲーションサーバは近接のグループ内のキャッシュサーバを選択する。選択されたキャッシュサーバは他のグループに対してキャッシュが存在するかどうか問い合わせる。問い合わせに対して他のグループはキャッシュの有無を返す。

このときキャッシュの存在場所により、キャッシュサーバの振舞が以下のように異なる。

1. キャッシュがどのグループにも存在しない。
2. キャッシュが自グループにしか存在しない。
3. キャッシュが他グループにしか存在しない。
4. キャッシュが自グループにも他グループにも存在する。

1 の場合、キャッシュサーバはリクエストをサーバに送信し、サーバからデータを受信する。そしてデータをキャッシュしてプロパゲーションサーバに送信する。2 の場合、キャッシュサーバはキャッシュが更新されているかどうかサーバに問い合わせる。キャッシュが最新であればその旨を示すメッセージを、更新されていればその内容をサーバから受信し、更新データをキャッシュしてプロパゲーションサーバに送信する。更新されていなければキャッシュデータをプロパゲーションサーバに送信する。3 の場合、キャッシュサーバはどのグループにリクエストを送信するかを選択する。選択されたグループはキャッシュが最新のものであるかどうかをサーバに問い合わせる。そのグループはキャッシュが最新であればその旨を示すメッセージを、更新されていればその内容をサーバから受信する。そしてリクエストを送信したキャッシュサーバにデータが送信される。そのキャッシュサーバはデータをキャッシュしてプロパゲーションサーバに送信する。4 の場合、選択されたグループが他グループであれば 3 の場合と同様にする。選択されたグループが自グループであれば 2 の場合と同様である。

4 グループ間の協調動作

このシステムにおいて、グループ間あるいは各グループとサーバを関係づける上で重要なことはキャッシュの一貫性を保つ、つまりオリジナルのデータとキャッシュされたデータが同じデータであるかをいかにして判断するかであり、3.2で行なわれるグループあるいはサーバの選択方法である。

4.1 キャッシュの一貫性

キャッシュの一貫性を保つために HTTP プロトコルの GET リクエストで使用される If-Modified-Since ヘッダを使用する。このヘッダは

If-Modified-Since: *date*

の形式で使用され (*date* は日時)、もし If-Modified-Since により指定された時刻以降に更新されていなければ、その旨をしめすメッセージ (304 Not Modified) が届き、更新されていれば、その内容が届く (200 GET)。キャッシュが存在する場合にこのヘッダを付加してサーバにリクエストすることによりキャッシュの一貫性を保つ。

4.2 グループ間メッセージフォーマット

グループ間で使用されるメッセージは TCP (Transmission Control Protocol) を使い送受信する。各グループへのキャッシュの有無を問い合わせるメッセージは

search *url*

の形式で行なう。search は文字列、*url* は URL である。

キャッシュのヒットメッセージは

hit *url servername last-modified-date content-length bandwidth delay*

ミスメッセージは

miss *url*

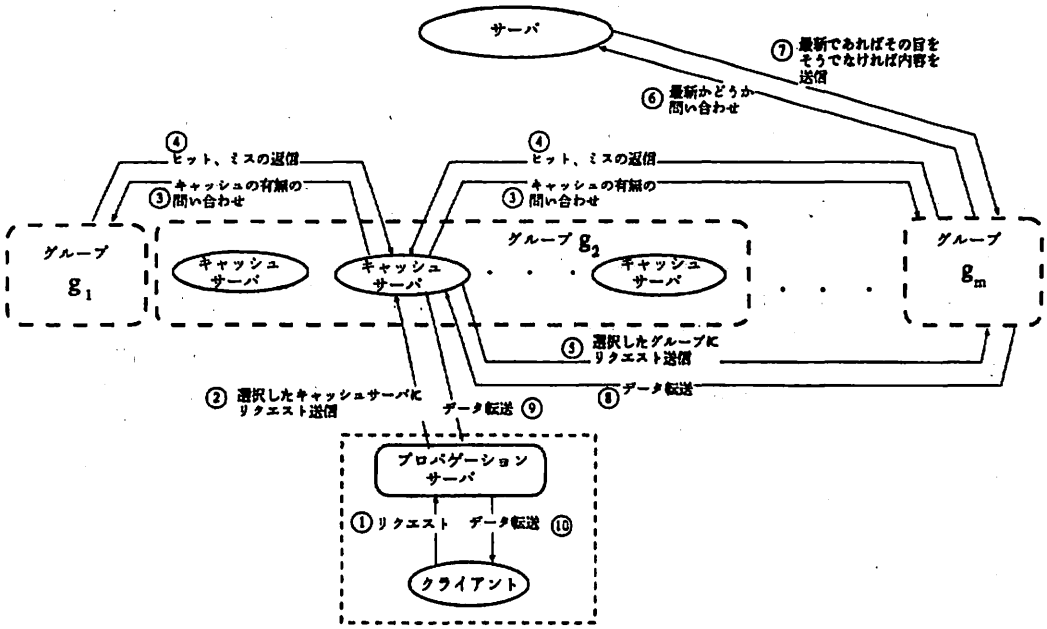

図 2: システム概略図

の形式で行なう。hit、miss は文字列、*url* は URL、*servername* はキャッシュを持っているサーバ名、*last-modified-date* は HTTP プロトコルのヘッダ部分 Last-Modified に記述してある最終更新日時、*content-length* は HTTP プロトコルのヘッダ部分 Content-length に記述してある内容の大きさ (byte)、*bandwidth* はそのグループとサーバの間のバンド幅 (byte/sec)、*delay* はそのグループとサーバの間の遅延時間 (s) である。

4.3 ネットワーク的距離の導入

複数のキャッシュサーバに目的とするデータがキャッシュされている場合、どのサーバからデータを受け取るかを選択する必要がある。Squid ではクライアントが指定したキャッシュサーバが、その neigbhor と parent にキャッシュデータがあるかどうか問い合わせ、最も速くキャッシュデータが返信されたキャッシュサーバからデータを取得する。提案システムでは、最新のデータにアクセスすることを方針としているため、すべてのサーバグループからのキャッシュヒットメッセージを一定時間待ち、ヒットしたグループのうちデータの最終更新日時が最新のものを採用する。さらに、その候補が複数ある場合には、返信したキャッシュサーバまでのネットワーク的距離を別に求めておき、それが最短であるものを採用する。これにより、クライアントに最新データを最短時間で転送できる。

4.3.1 バンド幅及び遅延時間の測定 [6]

ネットワーク距離の計測には UNIX コマンドの ping を用いる。ping は ICMP プロトコルの ECHO 要求 (echo request) を送信して、指定されたホストまたはネットワーク・ゲートウェイから ICMP の ECHO 応答 (echo reply) を受信することにより、データグラムの喪失の統計情報を検出するコマンドである。計測のモデルを図 3 に示す。計算機 S から計算機 G にむけて ping コマンドを実行しラウンドトリップ時間 (round-trip time) を得る。

バンド幅、遅延時間を算出するため T、T_1、T_2、b、L を以下ののように定義する。

T : 計算機 S から計算機 G へ L バイトのデータを

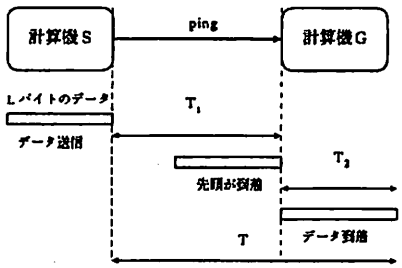

図 3: 計測モデル

送信するのにかかる総時間 (s)。

T_1 : L バイトのデータを計算機 S が送信してからデータの先頭が計算機 G に到着するまでの遅延時間 (s)。

T_2 : L バイトのデータの先頭が計算機 G に到着してからデータが全て到着するまでの時間 (s)。

b : 計算機 S と計算機 G の間のバンド幅 (byte/sec)。

L : パケットサイズ (byte)。

以上のような定義により、下の 2 式が成り立つ。

$$T_1 + T_2 = T \tag{1}$$

$$T_2 = L/b \tag{2}$$

式 (1)、(2) より遅延時間 T_1 を計算する式が導かれる。

$$T_1 = T - L/b \tag{3}$$

式 (1)、(2) に対して、パケットサイズの異なるデータ L、L' $(L > L')$ とそれぞれの送信総時間 T、T' を用いることにより、バンド幅 b を計算する式が成り立つ。

$$b = \frac{L - L'}{T - T'} \tag{4}$$

パケットサイズ $L = 1024$ バイト、$L' = 56$ バイトを ping コマンドにより午前 0 時からはじめて 3 時間おきに午後 9 時まで各時間毎に 10 回、計 70 回各計算機 G に送りラウンドトリップ時間を計測する。T、T' として 1 日の最小値を用い、式 (4) に代入してバンド幅を、求めたバンド幅より式 (3) により遅延時間 T_1 を計算する。これにより、ネットワークのバンド幅をほぼ正確に推定できることが確認されている。

4.3.2 グループの選択方法

予め、各グループとサーバ相互のバンド幅、遅延時間は 4.3.1の方法で測定されているものとする。グループ数を m、リクエストされたサーバを s、$M = \{1, ..., m\}$ とし、$i \in M$、$j \in M$、$i \neq j$ を満たす任意の i、j に対して次のように定義する。

g_i : 各グループを示す

b_{ij} : グループ g_i からグループ g_j 迄のバンド幅

T_{ij} : グループ g_i からグループ g_j 迄の遅延時間

b_{is} : グループ g_i からサーバ s 迄のバンド幅

T_{is} : グループ g_i からサーバ s 迄の遅延時間

L_{group} : L は問い合わせの際交換されるメッセージの最大長

L_{ifmod} : **If-Modified-Since** ヘッダを使用して **GET** リクエストを送信した時のメッセージ長

L_{notmod} : **304 Not Modified** メッセージの大きさ

L_{get} : クライアントからの **GET** リクエストメッセージの大きさ

L_{cont} : 内容を含んだデータの大きさ

プロパゲーションサーバがグループ $g_k (k \in M)$ を通してリクエストしたと仮定する (図 4)。この時、グループ g_k でのキャッシュの有無の問い合わせに対するタイムアウト時間を $j \in M$、$j \neq k$ を満たす任意の j に対して

$$T_{timeout} = 2 \times \max_j \{T_{kj} + L_{group}/b_{kj}\} \tag{5}$$

とする。$T_{timeout}$ 内に返信がない場合はキャッシュミスとみなす。

グループあるいはサーバを選択する際には、

- last-modified-date に記述された日付が最新のものが最も **304 Not Modified** のメッセージが届く確率が高いこと

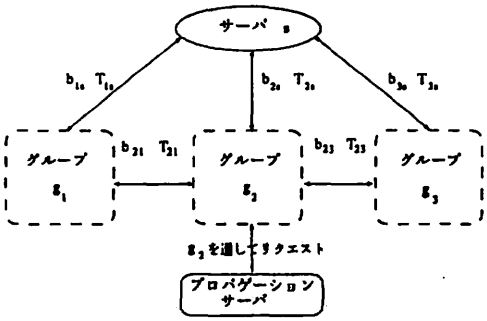

図 4: $m = 3$、$k = 2$ の場合の例

- ユーザから見たデータの到着時間を考慮に入れること

を念頭に置き、以下のような順序でグループあるいはサーバを選択する。

1. last-modified-date に記述された日付が最新であるものを選択
2. 日付が最新のものが 2 つ以上存在する場合、そのグループの集合を G_{hit} とすると $j \in G_{hit}$、$j \neq k$ を満たす任意の j に対して
   - グループ g_k にキャッシュがない場合は式 (6) により評価し選択

$$\min_j \quad (L_{ifmod} + L_{notmod})/b_{js} + 2T_{js} + (L_{get} + L_{cont})/b_{kj} + 2T_{kj}\} \quad (6)$$

   - グループ g_k にキャッシュがある場合は式 (7) により評価し選択

$$\min_j \quad \{(L_{ifmod} + L_{notmod})/b_{ks} + 2T_{ks}\ , (L_{ifmod} + L_{notmod})/b_{js} + 2T_{js} + (L_{get} + L_{cont})/b_{kj} + 2T_{kj}\} \quad (7)$$

3. 式 (6)、(7) により評価できない場合 (b_{ks}、T_{ks} あるいは b_{js}、T_{js} がわからない場合)
   - グループ g_k にキャッシュがない場合は $j \in G_{hit}$、$j \neq k$ を満たす任意の j に対して式 (8) により評価し選択

$$\min_j\{(L_{get} + L_{cont})/b_{kj} + 2T_{kj}\} \quad (8)$$

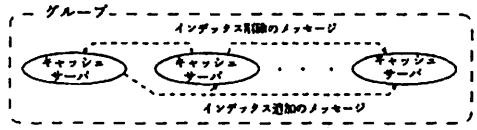

図 5: インデックス情報の更新手続き

- グループ g_k にキャッシュがある場合は g_k を選択

5 グループ内での協調

グループ内では、キャッシュサーバが個々のキャッシュを共有するような方法で協調させ 1 つの大きなキャッシュを持つようにする。そして同じ URL に対してキャッシュが重複しないようにする。そのため URL、キャッシュサーバのホスト名、最終更新日時、内容の大きさを記述した同じインデックスファイルを各キャッシュサーバが所有する。従って、グループ内で同じインデックスをもつための機構が必要になる。

プロパゲーションサーバでは、選択したキャッシュサーバにキャッシュがなく他のキャッシュサーバにある場合の対処方法が必要になる。

5.1 インデックス情報の更新

インデックス情報の更新手続きには、ある URL のデータをキャッシュしたという情報、つまり追加メッセージと、ある URL のキャッシュを消したという情報、つまり削除メッセージがある (図 5)。

複数のキャッシュサーバで同じインデックスを持つ、つまりインデックスファイルの一貫性を保つために、同じ URL では新しいインデックス追加あるいは削除メッセージを優先する方法をとる。この方法によりインデックス情報の送信の際、全てのキャッシュサーバをロックする必要がなくなる。図 6 のように同じ URL に対してインデクッス追加のメッセージが届くとその都度インデックスファイルの更新を行う。またインデックス削除のメッセージが届いたときにもその都度インデックスファイルの更新を行う。

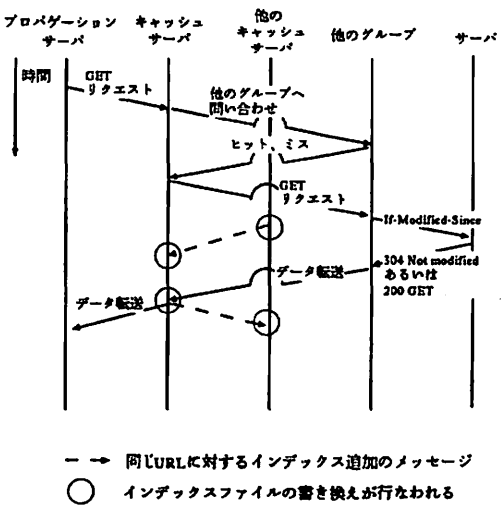

図 6: インデックス情報の更新

5.2 インデックス更新に用いるメッセージフォーマット

各キャッシュサーバ間で交換されるインデックス情報は UDP(User Datagram Protocol) を用いる。他のキャッシュサーバに送信するインデックス追加メッセージは

add *url servername last-modified-date content-length*

の形式で、インデックス削除のメッセージは

delete *url servername*

の形式で行なう。add、delete は文字列、その他は 4.2 と同様である。

5.3 プロパゲーションサーバ

プロパゲーションサーバは、クライアントとキャッシュサーバの間に位置し、キャッシュサーバの選択、データの中継を行なう。クライアントは従来のキャッシュサーバを指定する方法と同様にしてプロゲーションサーバを指定する。

クライアントからのリクエストがあるとプロパゲーションサーバは前回の履歴を用いてキャッシュサーバを選択する。つまり、前回リクエストされたところは再度リクエストされる可能性が強いため、同じドメイン名が再度リクエストされた場合前回アクセスされたキャッシュサーバを再度選択する。前回と異なったドメイン名がアクセスされた場合は乱数によりキャッシュサーバを選択する。そして GET リクエストをキャッシュサーバに送信する。リクエストされたデータを受信したらその内容をクライアントに送信する。

5.3.1 他のキャッシュサーバにキャッシュが存在する時の差し戻し方法

もし他のキャッシュサーバにキャッシュが存在すれば、400 Bad Request を利用したその旨を示すメッセージをキャッシュサーバがプロパーゲーションサーバに送信する。このメッセージの"Bad Request"の部分にキャッシュサーバのホスト名を記述する。

(例) HTTP/1.0 400 jupiter.ipc.hiroshima-u.ac.jp

これと同時に他のヘッダ、エラー内容も送信される。そしてこのホスト名のキャッシュサーバに GET リクエストを送信する (図 7)。

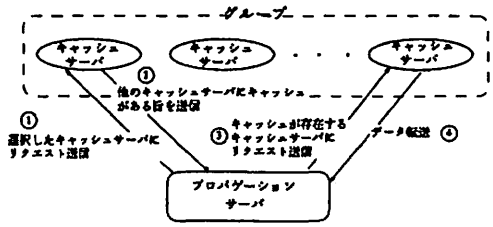

図 7: 他のキャッシュサーバにキャッシュがある場合

5.3.2 インデックス更新に伴う 3 度選択の回避

5.3.1 の差戻しが起こった場合、プロパゲーションサーバはキャッシュが存在するキャッシュサーバにリクエストを送信するが、そのキャッシュサーバでは何らかの理由でインデックスの内容が異なり、他のキャッシュサーバにキャッシュが存在するという情報である可能性がある。これが何度も起きるようなことがあれば、最悪の場合データをクライアントが取得できなくなる。従って、何度もリクエストが行なわれないように、HTTP プロトコルのヘッダ Via を利用する。このヘッダはリク

エストにおけるユーザエージェントとサーバの間、あるいはレスポンスにおけるサーバとクライアントの間に介在するプロトコルと受信者を示すためにゲートウェイやプロキシにより使用される。

(例) Via : 1.0 sekki 1.0 jupiter 1.0 mars

ここで 1.0 は HTTP プロトコルのバージョン、sekki、jupiter、mars はホスト名である。例のようにこのヘッダにはキャッシュサーバのホスト名が記述できるので同じグループ内のキャッシュサーバのホスト名が 2 度記述されていれば、2 度選択されたものとして他のグループあるいはサーバに GET リクエストを行なう。

6 おわりに

本論文では、WWW 分散キャッシュサーバとして複数のキャッシュサーバをグループ化し、グループ間ではキャッシュの重複を許すことにより負荷分散させ、グループ内ではキャッシュサーバを協調させ、キャッシュサーバ全体として強靭なシステムを提案した。

このシステムの実装及び評価が今後の課題として残る。また、本論文での提案では前回の履歴によりキャッシュサーバを選んでいるが、プロパゲーションサーバがキャッシュの履歴を持ち、適当なキャッシュサーバを選択することにより、キャッシュサーバの切り替えが起こりにくくするような拡張が必要であろう。

参考文献

[1] A. Loutonen and K. Altis, "World-Wide Web Proxies," Proc. WWW '94 Conference, Apr. 1994. (http://www.w3.org/hypertext/WWW/Proxies/)

[2] 佐藤 豊, "多目的プロトコル中継システム DeleGate," 電子技術総合研究所彙報 59, Jun. 1995. (ftp://etlport.etl.go.jp/pub/DeleGate/ETL-BULLTIN-95-06.ps.gz)

[3] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrel, "A Hierarchical Internet Object Cache," Tech. Rep. 95-611, University of Southern California, Dept. of Computer Science, Los Angeles, California, Mar. 1995. (ftp://ftp.cs.colorado.edu/pub/terchreports/schwartz/HarvestCache.ps.Z)

[4] R. Manlpani, J. Lorch, and D. Berger, "MAKING WORLD WIDE WEB CACHING SERVERS COOPERATE," Proc. 4th International WWW Conference, World Wide Web Journal, pp.107-117, Dec. 1995.

[5] D. Wessels, "Intelligent Caching for World-Wide Web Objects," Master's thesis, University of Colorado, Boulder, Colorado, Jan. 1995. (http://itp-www.colorado.edu/~wessels/Proxy/wessels-thesis.ps)

[6] 鳥谷 明雄, 西村 浩二, 相原 玲二, "インターネット上の距離空間とその応用," JAIN-OLU 合同シンポジウム 論文集 pp.75-80, Jan. 1995.