分散WWWキャッシュシステムの構成自動設定機構

井上博之[†,☆] 坂本岳史[†,☆☆]
山口英[†] 尾家祐二[††]

World Wide Web（WWW）によるネットワークのトラヒックの増加や，アクセス集中によるサーバの過負荷，それらにともなう利用者からみた応答時間の増大に対処するために，分散WWWキャッシュシステムが導入されている．現在のシステムでは，キャッシュサーバ群の構成情報に関する設定は各キャッシュサーバの管理者が自身の経験や知識をもとに手動で行っている．実際には，ネットワークや他組織のキャッシュサーバの構成がつねに変化するため，適切な構成を保つことは困難である．本論文では，分散WWWキャッシュシステムにヒントサーバを導入することにより，分散キャッシュシステムの設定・管理の自動化・簡略化を実現する方式を提案する．ヒントサーバは各キャッシュサーバの状態とそのキャッシュ内容を把握し，キャッシュサーバからのキャッシュ内容の問合せに応答する．提案システムを実装し評価を行った結果，キャッシュサーバの構成の変化に対応して構成情報の管理が自動化されたことを確認した．また，提案システムのキャッシュ全体のヒット率が従来システムを理想的な設定で動作させた場合と同一になることを確認した．

Automatic Configuration Mechanism for Distributed WWW Cache System

HIROYUKI INOUE,[†,☆] TAKESHI SAKAMOTO,[†,☆☆] SUGURU YAMAGUCHI[†] and YUJI OIE[††]

A distributed cache system for the WWW infrastructure is introduced that reduces both network traffic and the page retrieval latency. In this type of system, administrators of cache servers have to manually set up how and which neighboring caches should be incorporated, based on their experience and expertise. However, it is difficult for the administrators to update the related management information in response to network or neighboring cache server state changes in a dynamic and timely manner. In this paper, we propose a new WWW cache architecture with "hint server" which automatically updates the configurations associated with neighboring cache servers. In order to exchange the cache information between cache servers and the hint server, we extend the ICP (Internet Cache Protocol) message to contain the cache status and the detailed information about the object requested by a user. Through the deployment of this cache system including hint server, we conclude that the configuration of the cache system is automatically managed and the hit rate of the cache is equivalent to the ideal system.

1. まえがき

現在もWorld Wide Web（WWW）に代表される広域ネットワーク上での情報サービスは急速に発展を遂げている．その結果，ネットワークのトラヒックの増大，アクセスの集中によるサーバの過負荷，それらにともなう利用者からみた応答時間の増大が深刻な問題となっている．これらを解決するために，WWWキャッシングプロキシサーバ（キャッシュサーバ）[1]が広く利用されている．キャッシュサーバはプロキシサーバにキャッシュ機能を持たせたもので，クライアントが要求したオブジェクトをキャッシュ内に保有している場合には，そのキャッシュから必要なオブジェクトを取得しクライアントに転送する．必要なオブジェクトがキャッシュにない場合は，本来のオブジェクトを

† 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
☆ 現在，住友電気工業株式会社システムエレクトロニクス研究開発センター
Presently with Sumitomo Electric Industries, Ltd.
☆☆ 現在，松下電器産業株式会社
Presently with Matsushita Electric Industrial Co., Ltd.
†† 九州工業大学情報工学部
Department of the Computer Science and Electronics, Kyushu Institute of Technology

提供するWWWサーバからオブジェクトを取得し，クライアントに転送するとともに自身のキャッシュにも格納する．これにより同一オブジェクトの再利用が実現され，ネットワークのトラヒック量の削減と利用者に対する応答時間の短縮を実現する．

また後述するように，複数のキャッシュサーバを連携させ，全体として1つの大きなキャッシュサーバを構成する分散キャッシュシステムが実用化されている．このシステムでは，クライアントから要求を受けたキャッシュサーバが必要なオブジェクトを保存していない場合，他のキャッシュサーバにそのオブジェクトの有無を問い合わせ，オブジェクトを保持しているキャッシュサーバからオブジェクトを取得しクライアントに転送することにより，全体としてキャッシュの利用率を向上している．このシステムでは，各キャッシュサーバの管理者は，ネットワークの構成や他のキャッシュサーバの位置情報をもとに参照の対象となるキャッシュサーバを決定し手動で静的に設定しているのが現状である．そのため，ネットワークの状態や他のキャッシュサーバの変化に動的に対応できず，管理者につねに負担がかかるだけでなく，適切な設定を保つことが困難であるという問題点がある．

本論文では，ネットワークや各キャッシュサーバの状態およびその内容等を把握するヒントサーバを置くことにより，分散キャッシュシステムの設定・管理を自動化する手法を提案する．クライアントからの要求を受けたキャッシュサーバは，ヒントサーバに対して必要なオブジェクトの所在を尋ねる．ヒントサーバは自身が持つデータベースを検索し，必要なオブジェクトを保持しているキャッシュサーバを発見できた場合には，要求を出してきたキャッシュサーバにヒントとして与える．ヒントを受け取ったキャッシュサーバはヒントをもとにオブジェクトを取得するサーバを決定する．すなわち，各キャッシュサーバにヒントサーバの位置情報を設定するだけで分散キャッシュシステムを構成することが可能となり，各組織のキャッシュサーバの設定の自動化が実現できる．また，キャッシュサーバは本来の機能であるHTTP[2]データの転送処理やキャッシュの読み書きの処理のために負荷が常時高いにもかかわらず，他からのキャッシュ内容問合せに対する処理も行う必要がある．ヒントサーバではキャッシュ内容の通知および問合せの処理だけを行えばよいため，問合せを受けたときの応答時間がキャッシュサーバに比べて短くかつ安定することも期待できる．

評価はヒントサーバを導入した分散キャッシュシステムを実装して行う．自動設定の動作の確認はキャッシュサーバの構成を動作中に変化させた場合の挙動を観察することにより行う．また，ヒントサーバを導入したシステムにおける系のヒット率，トラヒック量などを従来システムの理想的な場合と比較し，システムの有効性を確認する．

2. WWW キャッシュサーバの構成情報の管理

2.1 分散キャッシュサーバ

最初に実用化されたキャッシュサーバはCERNによって開発されたWWWサーバ[3]のキャッシュ機能を使用したもので，WWWブラウザからみたプロキシサーバとして実装されていた．このようなプロキシとしてキャッシュサーバを参照する利用形態は現在でも変わっていない．Harvestプロジェクト[4]では，複数のキャッシュサーバを階層的あるいはメッシュ状に配置し全体として1つのキャッシュサーバを構成することで，キャッシュの利用効率を上げるようなシステムが開発された．その成果物をもとにしたキャッシュサーバが現在最も広く使用されているSquidキャッシュ[5]である．Squidでは，キャッシュ間の問合せのプロトコルとしてICP（Internet Cache Protocol）[6]を使用している．ICPはUDP（User Datagram Protocol）を利用しており，問合せおよびその応答の2種類のメッセージから構成されている．

以下では，階層的に配置されたキャッシュサーバ群において，利用者からみて上位の階層に位置するものを「親キャッシュサーバ」，同じ階層に位置するものを「隣接キャッシュサーバ」，本来のWWWオブジェクトを提供しているサーバを「オリジンサーバ」，利用者のWWWクライアントを「クライアント」と呼ぶ．

キャッシュ間の協調動作の機構としてICPを使った分散キャッシュシステム（図1）では，あるキャッシュサーバCS_iはクライアントあるいは別のキャッシュサーバから要求されたオブジェクトが自身のキャッシュにない場合，ICP問合せメッセージをあらかじめ管理者

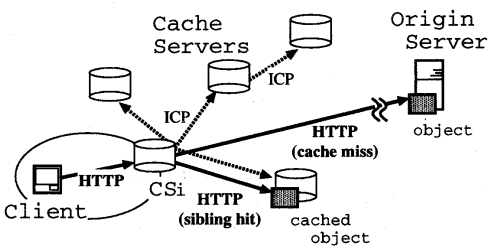


図1 ICPを使った分散キャッシュシステム
Fig. 1 Distributed cache system using ICP.

によって設定された隣接キャッシュサーバに対して送信し，その応答を待つ．CS_i はキャッシュにヒットした旨の応答メッセージを受け取った場合，最初にヒットを返した送信元のキャッシュに対してオブジェクトの取得を行う．また問合せを行ったすべての隣接キャッシュサーバからオブジェクトが存在しない旨（キャッシュのミス）のメッセージを受け取るか，あるいは受信のタイムアウト（通常1から2秒程度）が発生した場合は，CS_i はあらかじめ管理者によって記述された規則に従って親キャッシュサーバあるいはオリジンサーバに対してオブジェクトの取得を行う．

ICPのような協調動作のためのメッセージを使用しない分散キャッシュシステムも存在する．クライアントにおいてプロキシサーバ選択の動作が記述できること[7]を利用して，アクセスしようとするオブジェクトのURL[8]文字列をもとにキャッシュサーバに決定的にアクセスを行うことで分散処理を行う方式がある[9),10)]．これらの方式では，同一のURLで表されるオブジェクトは同じサーバにアクセスされるため，ICPを使用しなくてもキャッシュの利用率を維持したままでキャッシュサーバの分散が可能である．しかしながら，動作記述を設定できるブラウザは限られており，またキャッシュサーバ間の協調動作を指定できない．

2.2 分散キャッシュサーバの構成管理

現在の分散キャッシュシステムにおいては，サーバの構成情報の管理は各キャッシュサーバの管理者が経験をもとに手動で行っている．すなわち，初期設定時またはキャッシュサーバおよびネットワークの状態や構成が変化したときは，管理者が自身の経験や知識をもとに，キャッシュの動作が適切になるように構成情報を更新している．構成情報とは他のキャッシュサーバを参照するために必要な情報であり，具体的には，サーバのホスト名またはIPアドレス，ICPポート番号およびHTTPポート番号からなる．前述のSquidキャッシュでは，この記述はプログラムの設定ファイルとして記述されており，変更には管理者の介在とキャッシュプログラムの再起動が必要である．

たとえば，WIDEプロジェクトを例にみると，バックボーンの再構成のような大規模なネットワークの変更が過去1年間に5回行われた．また，WIDE内のキャッシュサーバの変更は多いときで3週間に1回程度の割合で行われた．ネットワークやキャッシュサーバの構成の変更が起こるごとに，各組織のキャッシュサーバ管理者はその情報をもとに構成情報の設定を変更しなければならなかった．さらに局所的なネットワークやキャッシュサーバの状態の変化はより頻繁に発生していたが，キャッシュサーバの管理者は特に大きな不具合がなければそのまま運用を行っていた．

2.3 従来の自動管理手法

このような構成情報の管理に必要な多くの時間を削減するために，いくつかの手法が提案されている．Squidキャッシュでは，ICPの問合せにIPマルチキャスト[11)]を利用することができ，その問合せは同じマルチキャストグループに参加したキャッシュサーバすべてに届く．よって，あるキャッシュサーバ群に加わるためには，そのグループに対応するマルチキャストアドレスを構成情報として設定するだけでよい．しかし，この方式ではグループに参加するキャッシュサーバの制限を行うことが難しく，さらにICPの応答メッセージはユニキャストで送信されるため，1つの問合せに対してサーバ台数分の応答メッセージの通信が発生するという問題点がある．

またSquidキャッシュでは，他のキャッシュから参照されるのに必要な情報をキャッシュ自身が定期的にアナウンスする仕組みもある[12)]．アナウンスされた情報はあるサーバに収集され，必要に応じてその一覧をWHOIS[13)]プロトコルを使用して取り出せるようになっている．よって，管理者はWHOISを使ってキャッシュサーバの一覧を取り出して構成情報として設定し，さらに定期的にWHOISで情報を得ることでキャッシュのICPおよびHTTPポート番号の変化に自動的に対応できる．しかし，その一覧からどのキャッシュを使用するかという判断は管理者に任されており，経験をもとに手動で設定する手間は従来と同じである．

2.4 構成情報の自動管理機構

既存のキャッシュシステムにおける以上のような問題は，あるキャッシュサーバが，他のサーバのキャッシュ内容およびその状態といった情報を持たないことから生じている．よって，解決のためにはサーバの状態やキャッシュ内容の変更を他のサーバに通知し，通知されたサーバはその内容を記憶し，適切なサーバを選択するようにすればよい[14)]．しかしながら，キャッシュサーバのプログラムはHTTPデータの転送処理，キャッシュディスクとの入出力処理などを行い，すでにコンピュータの資源を多く必要としていることから[15)]，さらに通知された内容を記憶することは負担が大きく実現が困難である．また，各キャッシュサーバで同一の通知内容を記憶することになり冗長である．

本論文では，各キャッシュサーバからその状態やキャッシュ内容の変更の通知を受信・記憶し，また他のキャッシュサーバからの問合せに答えるヒントサーバを導入したキャッシュシステムを提案する．ヒントサーバは

各キャッシュサーバのキャッシュ内に保持しているオブジェクトをつねに把握することにより，あるキャッシュサーバからの問合せ要求に対してオブジェクトをどのキャッシュサーバから取得すべきかという情報を応答する．本システムでは，各キャッシュサーバはヒントサーバに関する情報のみを設定すればよく，他のキャッシュサーバに関する構成情報の設定を行う必要がなくなる．また管理者がネットワークや他のキャッシュサーバの状態の変化を意識して設定を行う必要もなくなる．すなわち，キャッシュサーバの構成情報の管理を自動化・簡略化することができる．

ヒントサーバと同様のキャッシュ管理モデルとして，密結合型マルチプロセッサ構成におけるディレクトリ参照型キャッシュがある[16),17)]．密結合型マルチプロセッサにおけるキャッシュはメモリや他プロセッサからの応答時間がある一定時間で保証されており，またキャッシュの内容の一貫性にも厳密な保証が必要とされる．WWW キャッシュにおいては，オリジンサーバや他のキャッシュサーバからの応答や内容の一貫性は一般に保証されていない．さらにインターネットの到達保証性や帯域幅が変化するような通信路を使用するため，密結合型マルチプロセッサシステムと同様の議論は難しく，システムの有効性に関しては実装して評価を行う必要がある．

3. ヒントサーバを使った分散キャッシュシステム

3.1 システムモデル

先に述べた問題点を解決するために今回提案するキャッシュシステムでは，ヒントサーバを導入した点が既存のシステムと異なる．システムのモデルは**図2**のようになり，大きく分けて4つの部分すなわち，クライアント，オリジンサーバ，キャッシュサーバ，ヒントサーバから構成される．

ヒントサーバは各キャッシュサーバのキャッシュの内容を把握するために，キャッシュサーバからそのキャッシュの内容が変更された旨の通知メッセージを受け取り，自身の持つデータベースを更新することによって各キャッシュの内容との整合性を保つ．また，キャッシュサーバからの問合せメッセージを受け取るとデータベースを検索し，キャッシュサーバがクライアントから要求されたオブジェクトをどこから取得すべきかというヒント情報を応答メッセージとして返す．ヒント情報としては，オブジェクトを持っているキャッシュサーバに関する情報，あるいはオリジンサーバから直接取得すべきという情報のいずれかになる．後者の情報が返されるのは，いずれのキャッシュサーバもオブジェクトを持っていないときや，キャッシュサーバおよびオリジンサーバのネットワークの状態などから判断されたときである．問合せおよび応答のプロトコルには，既存の分散キャッシュサーバ間で使用されているICPに，ヒント情報を格納できるように拡張を行ったメッセージを使用する．

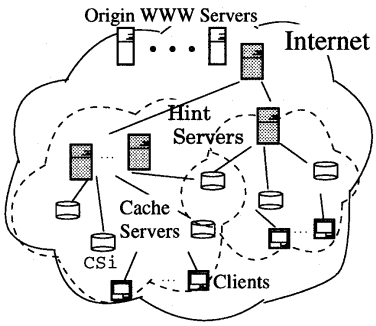


図2 システムモデル
Fig. 2 System model.

ヒントサーバを階層的に配置あるいは同じ階層で複数設置することで，スケーラビリティや冗長性を持った構成をとることが可能となる．階層的に配置された場合，キャッシュの変更通知メッセージや問合せメッセージは上位に位置するヒントサーバにも転送され処理される．また，あるキャッシュサーバが参照するヒントサーバを複数にすることで冗長構成となり，たとえば1台のヒントサーバからの応答がなくなった場合でも変わりなく動作できる．なお，ヒントサーバからの応答がまったくなくなった場合でも，キャッシュサーバは協調動作を行わない単体のキャッシュとして振舞うため，クライアントからみた影響は小さい．

各キャッシュサーバは自身のキャッシュ内にオブジェクトを追加・更新・削除した際に通知処理を行うが，通知に必要な処理は百数十バイトのUDPメッセージを生成し送信するだけであり，キャッシュサーバの本来の処理であるHTTPの転送処理やキャッシュディスクの読み書きの負荷に対して十分小さい[15)]．

キャッシュサーバはヒントサーバに問合せを行いその応答を待つことから，ヒントサーバの応答時間は十分短い必要がある．既存のキャッシュサーバにおけるICP問合せ時の応答時間は通信時間を除いて数msecから数十msec程度である．ヒントサーバからの応答がない場合は，キャッシュサーバは適当な待ち時間を設定し応答がタイムアウトするまで待つ．その場合，キャッシュサーバは自身のデフォルト時の動作設定（一

般にはオリジンサーバから直接オブジェクトを取得する，すなわち単体のキャッシュとして振舞うような設定）に基づいてオブジェクトの取得を行う．このような応答のタイムアウトは従来の分散キャッシュシステムでも同様に存在する．しかし，複数の異なるネットワーク上のキャッシュサーバからの応答を待つ必要があるため，応答時間にばらつきがあり，またタイムアウトが発生する確率も高いことが筆者らのキャッシュサーバの運用から経験的に得られている．本方式のようにヒントサーバからの応答のみを待つ場合ならば，応答時間のばらつきが減少することが期待できる．よって，過去のヒントサーバからの応答時間の履歴をもとに，タイムアウトまでの待ち時間をキャッシュサーバで予測し適切な値を設定することも可能になる．

3.2 システムの動作とプロトコル

3.2.1 ICP メッセージ

ヒントサーバとキャッシュサーバ間のメッセージには，既存の ICP メッセージを上位互換を保ちながら拡張したものを利用する．追加した ICP のメッセージは以下のとおりである．

- ヒント通知メッセージ（HNOTIFY）
  キャッシュサーバが，キャッシュの内容の追加あるいは削除されたことをヒントサーバに通知するために用いる．
- ヒント問合せメッセージ（HQUERY）
  キャッシュサーバが，クライアントから要求されたオブジェクトが自身のキャッシュにない場合にヒントサーバに問合せを行うために用いる．
- ヒント応答メッセージ（HREPLY）
  ヒントサーバが，受信したHQUERYに対して自身の持つデータベースから検索を行い，その結果をキャッシュサーバに通知するために用いる．

上記の拡張 ICP メッセージの一部が失われた場合は，キャッシュをミスと判断するか，あるいはタイムアウトによりデフォルトの動作となる．このことから，クライアントからみた影響は小さいと考えられるため，メッセージの再送処理は行わない．

3.2.2 プロトコル

提案したシステムの動作は以下のようになる．

キャッシュサーバ

キャッシュサーバ CS は，クライアント CL から HTTP によりオブジェクト obj の取得要求を受信した場合に以下の処理を行う（**図3**）．

(C-1) 自身のローカルキャッシュを検索する．

(C-2a) objが見つかった場合はobjをCLに転送し，処理を終了する．

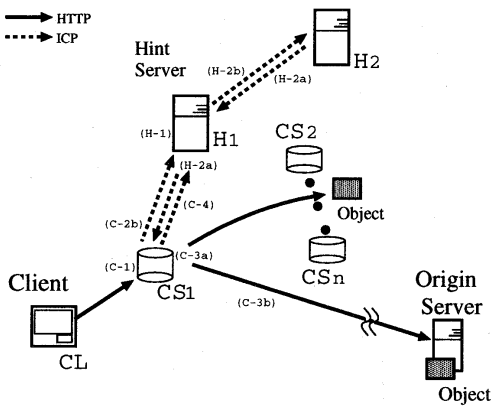


図3 提案システムにおけるWWWアクセス手順
Fig. 3 WWW access sequence on the proposed system.

(C-2b) objが見つからなかった場合は，ヒントサーバ H に HQUERY を送信し，HREPLY を待つ．

(C-3a) H から HREPLY を受け取ると，そのヒント情報に基づいて隣接キャッシュサーバあるいはオリジンサーバから obj を取得し，CL に転送する．

(C-3b) HQUERY を受信する前にタイムアウトが発生した場合は，オリジンサーバから obj を取得し，CL に転送する．

(C-4) 自身のキャッシュに obj を格納し，その旨を通知するため H に HNOTIFY を送信する．

また，キャッシュからオブジェクト obj を破棄した際には，その旨を通知するため H に HNOTIFY を送信する．

一定時間 H からの HREPLY を受信できない場合，CS はその H を利用できなくなったと判断し，以降の問合せを行わない．利用できなくなった H は一定時間ごとに HQUERY を送信し，HREPLY を受信したときに再び利用可能と判断する．

ヒントサーバ

ヒントサーバは自身のデータベースに情報を持たない状態で起動し，キャッシュサーバからの HNOTIFY をもとにデータベースを更新していく．

ヒントサーバ H はキャッシュサーバ（あるいは下位のヒントサーバ）CS からオブジェクト obj に関する HQUERY を受信した場合に以下の処理を行う（図3）．

(H-1) 自身の持つキャッシュ情報データベースを obj の URL をキーとして検索する．

(H-2a) 一致するものが見つかった場合は，HREPLY にヒント情報を格納し，CS に送信する．

(H-2b) 一致するものが見つからなかった場合は，上位のヒントサーバに対して HQUERY を送信し，HREPLY を待つ．上位のヒントサーバがない場合，あるいは応答のタイムアウトが発生した場合は，obj を持つキャッシュサーバが存在しない旨のヒント情報を CS に送信する．また，上位のヒントサーバから応答を受信した場合は，その情報を CS に送信する．

また，キャッシュサーバ CS から HNOTIFY を受信した際には，その情報をもとに自身のキャッシュ情報データベースを更新する．

3.2.3 キャッシュサーバの状態変化への追従

キャッシュサーバの状態把握

ヒントサーバ H は ICP を送信してきた CS の一覧をデータベース上に持ち，一定時間 CS からの ICP を受信しない場合，H はその CS に対して ICP を送信し CS が運用状態かどうか定期的に検査する．その結果，利用できなくなったと判断した場合，ヒント情報のキャッシュサーバ候補から除外する．再び CS の ICP を受信するか検査の結果利用可能と判断した場合には，キャッシュサーバ候補に登録する．

キャッシュサーバの起動時

キャッシュサーバの起動時に，自身の持つすべてのオブジェクトの情報をヒントサーバに HNOTIFY で送信する．

キャッシュサーバの停止時

キャッシュサーバ CS の停止時に，HNOTIFY でヒントサーバ H にサーバが停止する旨を通知する．そのメッセージを受信した H はその CS に関するデータベースをすべて削除する．

3.3 ICP メッセージの数とトラヒック

ICP メッセージで相互にキャッシュ情報の問合せを行っている何台かのキャッシュサーバからなるシステムの，ICP メッセージの数とそのトラヒックの指標を求める．

従来システムでは，ローカルなキャッシュでヒットしなかった場合，キャッシュサーバは問合せの ICP を他のキャッシュサーバに送り，受け取ったサーバはその応答の ICP を返す．ヒントサーバを使った提案システムでは，ローカルなキャッシュでヒットしなかった場合，キャッシュサーバは問合せの ICP をヒントサーバに送り，ヒントサーバはその応答の ICP を返す．また，他のキャッシュサーバにヒットしなかった場合，キャッシュサーバは自身のキャッシュにオブジェクトを保持し，その追加と削除の際にヒントサーバに対して通知の ICP を送る．

よって，従来システムと提案システムにおける ICP メッセージのそれぞれの総数 I_c，I_h は次式で与えられる．ここで，システムにおけるキャッシュサーバの数を n，ヒントサーバの数を h，クライアントからの総アクセス数を N，ローカルなキャッシュでのヒット率を a，隣接キャッシュでのヒット率を b，URL の平均長を L [byte] とした．

$$I_c = 2(1-a)(n-1)N$$

$$I_h = 2(1-a)hN + 2(1-a-b)hN$$

また，提案システムで使用する ICP のサイズについては，URL を除いた部分のデータが 20 byte[6] から 80 byte に増加しているため，それぞれのシステムにおけるトラヒック T_c，T_h [byte] は次式で与えられる．実際には，これに UDP/IP/データリンクの各ヘッダのオーバヘッドが加わる．

$$T_c = (20+L)I_c$$

$$T_h = (80+L)I_h$$

たとえば，$n=5$，$h=1$，$a=0.4$，$b=0.1$，$L=50$（図 6 (a)）の場合，$I_c = 5N$，$I_h = 2.2N$，$T_c = 350N$，$T_h = 286N$ となる．

4. 実装と評価

4.1 実　　装

提案したシステムを実装して評価を行った．開発したヒントサーバは C 言語で約 1,500 行からなり，一般的な Unix プラットフォームの上で動作する．現在動作を確認している OS は BSDI 社の BSD/OS2.1，SGI 社の IRIX6.3，Sun Microsystems 社の Solaris2.5 である．キャッシュサーバは，現在インターネットで広く利用されている Squid キャッシュ[5] をベースに変更を行った．具体的には，ヒントサーバへの問合せの処理の追加，ヒントサーバからのヒント情報を解釈しオブジェクトを取得する処理の追加，ヒントサーバにキャッシュの内容の変更を通知する処理の追加からなり，変更部分のコードは C 言語で約 700 行である．

キャッシュサーバとヒントサーバの間でやりとりされるメッセージには ICP を使用し，既存の ICP との互換性を維持するために Opcode を追加定義するようにした．新設した Opcode の ICP のフォーマットは**図 4** のようになり，オブジェクトの持つ属性やヒント情報を格納できるようになっている．RFC2186 Header と書かれた部分は既存の ICP と共通の部分である．新規に定義した Opcode は，`ICP_OP_HNOTIFY`，`ICP_OP_HQUERY`，`ICP_OP_HREPLY` の 3 つである．各 Opcode は Options 部分に格納されるサブコードを持

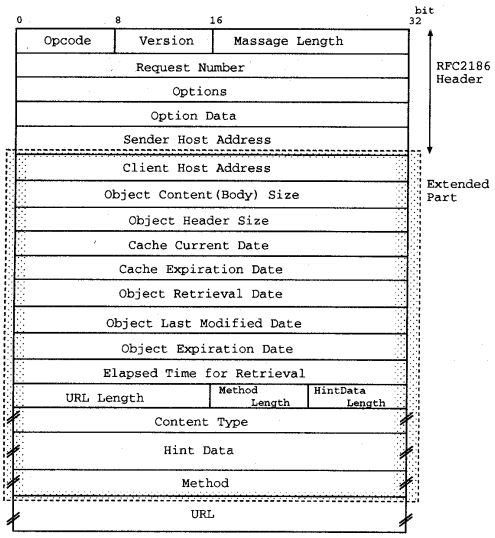


図4 追加したICPのフォーマット
Fig. 4 New ICP message format.

ち，ヒント情報の種別などを表すようになっている．また，トランスポート層のプロトコルとしては，既存のICPと同様にUDPを使用している．

ヒントサーバのキャッシュ内容情報データベースはURLをキーとしており，ここで扱う必要のあるオブジェクトの数は数万から数十万件になるため，検索を$O(logN)$で可能な木構造で実装した．データベースに格納される情報は以下のとおりである．

- オブジェクトのURLおよび属性（ヘッダやコンテンツのサイズ，最終更新時刻，キャッシュにおける有効期限など）．
- そのオブジェクトに関するヒント情報．
- キャッシュサーバの状態．

4.2 評　価

4.2.1 実験内容

今回提案した方式に基づくキャッシュシステムの評価として，従来の分散キャッシュシステムとの比較を行った．実験は，**図5**のように両システムの典型的な構成を作成し，クライアントには実際のキャッシュサーバで運用中に記録したアクセスログを入力してシステムの性能を計測した．アクセスログには奈良先端科学技術大学院大学で平日5日間の間に学内の利用者からアクセスされた内容のうち，キャッシュ可能なHTTPのメソッドであるGETリクエストを抽出したものを使用した．アクセスログは50万件分で，そのURLの長さの分布や平均長，およびオブジェクトのデータサイズとその分布は**図6**に示すとおりである．

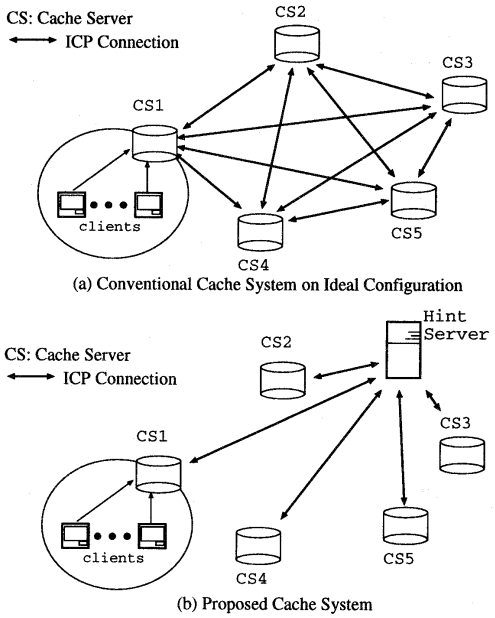


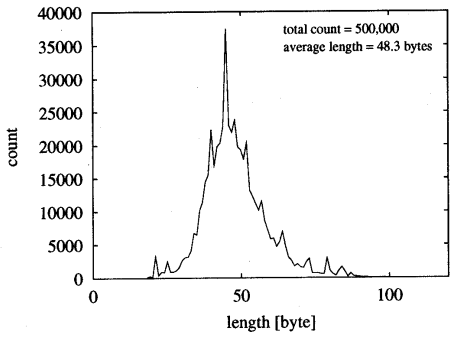
図5 実験システムの構成
Fig. 5 Experiment system configuration.

使用したキャッシュサーバの構成を**表1**に示す．

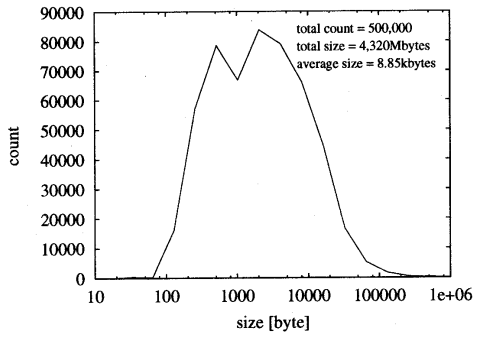
実験で比較した項目はそれぞれのシステムにおける，(1) キャッシュ構造定義の内容，(2) キャッシュサーバの構造に変化があったときに必要な管理者の対応内容，(3) システム全体としてみたときのキャッシュのヒット率，である．(1)，(2) は自動設定機構の動作の確認のためであり，(3) は提案方式と従来方式で理想的な設定を手動で行った状態を比較するためである．

実際の運用データを解析する手法ではなく，同一のアクセス記録を実験システムに入力する方法で評価を行った理由は，ネットワークやキャッシュサーバの利用状態が刻々と変化する運用システムでは，ヒット率を同じ条件で比較するのは難しいと判断したためである．ただし，この評価においても，オリジンサーバとそこに至るまでのネットワークの状態はそれぞれの実験でまったく同じではないため，実験ごとに同じ結果が得られるとは限らない点に注意しなければならない．

なお，今回実験を行ったキャッシュシステムにおけるサーバの数は5台とした．実際にWIDEプロジェクトで運用中の11の組織のキャッシュサーバを調査したところ，定義されている他のキャッシュサーバへの参照数が3から8台であったためである．また，一般にあまり多くのサーバを参照するとキャッシュミス時におけるICPの応答待ちのタイムアウトの確率が増



(a) URL Length Distribution



(b) HTTP Data Size Distribution

図6 実験に用いたアクセスログの統計情報
Fig. 6 Access log statistics for experiment.

表1 キャッシュサーバの構成
Table 1 Configuration of a cache server.

name	value
Model	SGI O2
OS	IRIX6.3
Phyical memory	96 Mbytes
Network I/F	100Base-TX
WWW cache memory	8 Mbytes
WWW cache disk	70 Mbytes
ICP timeout	1 second

え応答時間の悪化につながる．よって，参照数は数台程度に抑えておくことがインターネットでは推奨されており，実験における5台という数は適切と考えた．

4.2.2 実験結果

(1) キャッシュの構造定義

各キャッシュサーバにおける，キャッシュの構造定義の内容（抜粋）は**図7**のようになる．従来のシステムについてはcache2の設定を例として示したが，実際はそれぞれのキャッシュサーバで異なっている．

従来の分散キャッシュシステムでは，cache1からcache5までの各キャッシュサーバにおいてすべて設定

```
#          hostname   type      HTTP/ICP port

cache_host cache1   sibling    3128 3130
cache_host cache3   sibling    8000 8130
cache_host cache4   sibling    3128 3130
cache_host cache5   sibling    8888 3130
```

(a) Conventional Cache System on Ideal Configuration

```
#          hostname   type      HTTP/ICP port

cache_host hints1 hintserver     0 4649
```

(b) Proposed Cache System

図7 キャッシュ構成情報の比較
Fig. 7 Comparison of configuration of cache structure.

が異なり，他のキャッシュサーバを参照するための情報を管理者があらかじめ入手し，それを列挙しなければならなかった．提案した方式では，すべてのキャッシュサーバの設定はヒントサーバの情報のみを記述するだけで，各キャッシュサーバごとに異なる設定を行う必要はなかった．

(2) キャッシュサーバの変化時の対応内容

図5の構成において，ある1台のキャッシュサーバが停止（そこまでのネットワークが停止した状態を含む）した場合の挙動を調べた．従来のシステム(a)においては，停止したサーバからICPの応答が送られないため，他のサーバでICPの問合せ処理においてタイムアウトが発生した．また，残りの4台のサーバからなる構成に変更するためには，各サーバの設定を手動で変更しプログラムを再起動する必要があった．提案システム(b)においては，ヒントサーバが停止したキャッシュサーバを自動的に検出し，ヒントサーバが返すICPの中のヒント情報から当該キャッシュサーバが除外された．すなわち，残りの4台のキャッシュサーバどうしで協調動作する状態へ自動的に移行し，各キャッシュサーバの設定変更は不要であった．

次に，図5の構成にキャッシュサーバを新規に1台追加した場合の挙動を調べた．従来システム(a)においては，新設のサーバは既存の5台を参照するように手動で設定しなければならず，またすべてのサーバすなわち6台で協調動作させるためには，各サーバの設定変更とプログラムの再起動が必要であった．提案システム(b)においては，新設のサーバの設定ファイルは既存のキャッシュサーバと共通でよく，また新設のサーバからヒントサーバへキャッシュの状態が通知されると，ヒントサーバはその情報を各キャッシュサーバからの問合せの応答に含むようになった．すなわち，新設のサーバを含めた6台での協調動作を自動的に行うことができた．

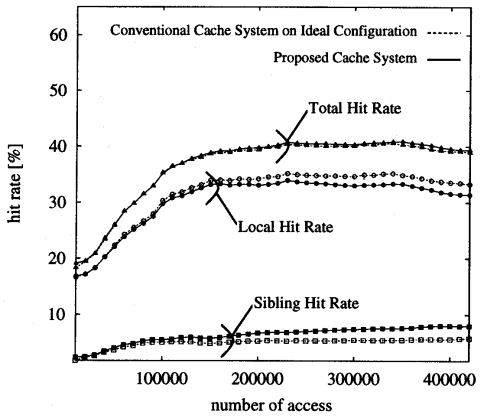


図8 キャッシュヒット率の比較
Fig. 8 Comparison of cache hit rate.

(3) キャッシュのヒット率

分散キャッシュシステムにおいては，システム全体のヒット率は各キャッシュサーバのローカルなヒット率と，ICPを使った他のキャッシュサーバあるいはヒントサーバへの問合せによるヒット率の和で表される．図5の構成で，従来システムと提案システムの各場合におけるローカルなヒット率（Local Hit Rate）とICPによる隣接キャッシュへのヒット率（Sibling Hit Rate）およびそれらの和（Total Hit Rate）を測定したものは**図8**のようになった．なお，図5における，(a)の従来システムのICP参照形態はフルメッシュであり，ICPによるヒット率が最大になる理想的な構成となっている．この状態で，提案システム(b)の総ヒット率は理想的な設定状態(a)のものと同一になることが確認できた．

また，隣接キャッシュサーバへのヒット率は提案システムのほうが従来システムを上回っており，この原因として後述のICPのタイムアウト率の低さが影響していると考えられる．なお，各キャッシュサーバのキャッシュ容量は有限であるため，系全体としてのヒット率は理想的な構成を超えることはないと考えられる．

4.3 考　　察

4.3.1 自動設定機構の実現

ヒントサーバを導入した提案システムにおいて，4.2.2項の結果（1）より，分散キャッシュシステムの構成情報の管理の自動化を確認することができた．結果（2）より，各キャッシュサーバはヒントサーバの情報のみを最初に設定することで，サーバの増減に対して自動的に対応できた．また，結果（3）より，システム全体のヒット率は従来の分散キャッシュの理想的な構成の場合とほぼ同一であり，提案システムの有効性が確認できた．

表2 ICPの数とトラヒックの比較
Table 2 Comparison of ICP messages and the traffic.

	Conventional System	Proposed System
Total ICPs	270×10^4	129×10^4
ICP Traffic		
Total	148 mps (189 MB)	122 mps (168 MB)
[H]QUERY	79.0 mps (94.6 MB)	32.2 mps (44.4 MB)
[H]REPLY	79.0 mps (94.6 MB)	32.2 mps (44.4 MB)
HNOTIFY	— (—)	57.4 mps (79.1 MB)
Local Hit Rate	32.4%	30.5%
HTTP Traffic	1,100 MB	1,010 MB
Sibling Hit Rate	5.5%	7.5%
HTTP Traffic	150 MB	224 MB
Miss Rate	62.1%	62.0%
HTTP Traffic	3,070 MB	3,080 MB

mps: messages per second, MB: Mbytes

4.3.2 ICPのメッセージ数とトラヒック

表2に，従来システムと提案システムにおけるICPメッセージ数とそのトラヒックを3.3節の指標と実験結果をもとに計算した値を示す．また，キャッシュサーバにヒットした場合とミスした場合の割合とそれぞれのHTTPによるトラヒックの結果も同時に示した．

提案システムでは，総ICP数が半分以下になり，バイト数ベースのトラヒックも1割程度減少している．バイト数ではあまり違いがない理由は，提案システムで拡張したICPのヘッダ部分が大きいためであると考えられる．今回実装した拡張ICPのフォーマットではヘッダの各フィールドが固定で割り当てられているためメッセージごとに未使用の部分があるから，メッセージの種類に応じてヘッダを定義することでトラヒックを減らせる可能性がある．

また，HTTPによるトラヒックと比較すると，ICPによる総トラヒックは約4%であった．これは隣接キャッシュサーバにヒットした際のHTTPトラヒック（表のSibling HitのHTTP Traffic）と同程度に相当することが分かった．

4.3.3 ヒントサーバの性能

ヒントサーバ自体の性能を評価するために，ヒントサーバのICP問合せに対する応答時間の計測を行った．同時に，性能の低いコンピュータでの応答時間も比較できるように，ワークステーション（SGI O2; CPU R5000 180 MHz; Memory 96 MB）とPC Unix（BSD/OS; CPU 486DX2 66 MHz; Memory 32 MB）の2台のマシンでヒントサーバを動作させ，キャッシュサーバから同じICPメッセージを送って応答時間を計測した．ヒント問合せメッセージに対する応答時間を**図9**に示す．ワークステーションにおける応答時間

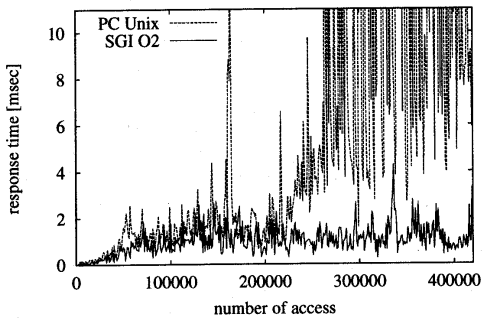


図9 ヒントサーバのICP応答時間
Fig. 9 ICP response time of a hint server.

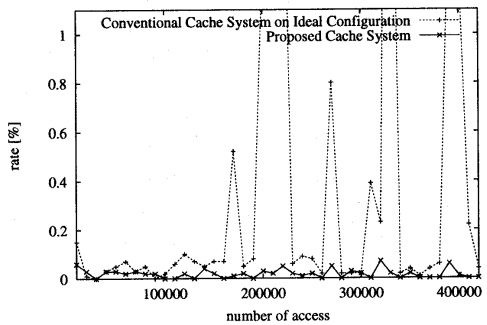


図10 ICPのタイムアウト率
Fig. 10 ICP time-out rate.

は，1から2msecの間で安定していることが分かる．PC Unixも3msec以内にほぼ収まっているが，20万アクセスを超えるころから応答時間が大きくなっている．これはマシンの主記憶が少なく仮想記憶のスワップを起こしていたためであった．

この結果から，実装したヒントサーバの応答時間は2msec程度であり，これはWWWのアクセス時間からみて十分に短い応答時間である．またヒントサーバは各キャッシュの内容を記憶しておくための主記憶が十分にあれば性能の低いコンピュータでも十分動作することも分かった．また，キャッシュの1つのオブジェクトの情報を記憶するのに必要な主記憶の大きさは平均で220byteであった．たとえば，10万のオブジェクトを記憶する場合で主記憶は約22MB必要となる．

4.3.4 ICPのタイムアウト率

実験を行った図5の両システムにおいて，キャッシュサーバからのICP問合せ時のタイムアウトの発生割合を図**10**に示す．なお，表2に示すとおり，従来システムにおける各キャッシュサーバでのICPメッセージの平均受信数は毎秒30個（表のトラヒック値は5台の合計である）で，提案システムにおけるヒントサーバでの平均受信数は毎秒90個であった．ヒントサーバからの応答はほとんどタイムアウトが発生していないが，従来システムではキャッシュサーバどうしのICP問合せ時のタイムアウトが高い率で発生していることが分かる．文献15)にあるようにキャッシュサーバ自体の負荷がかなり高いため，クライアントからのアクセスが集中するとキャッシュサーバにおけるICPの応答処理が遅れ，結果としてタイムアウトを発生させているものと考えられる．一方，ヒントサーバはICPメッセージの処理のみを行えばよいため，受信ICPメッセージ数が多いにもかかわらず安定した応答時間が得られたと考えられる．

5. むすび

本論文では，分散WWWキャッシュシステムの設定の自動化の支援機構を実現するために，ヒントサーバ付き分散キャッシュシステムを提案した．ネットワークやキャッシュサーバの状態，また各キャッシュの内容等を把握するヒントサーバを導入することで，従来の分散WWWキャッシュシステムでは困難であったシステムの設定・管理の自動化を実現できた．また，自動化した場合でも系全体のヒット率は理想的な設定を手動で行った場合と同一であることを確認し，隣接キャッシュへのヒット率が向上することが得られた．同時に，ICPメッセージ数の削減，応答待ちタイムアウトの発生割合を減少させることが可能となった．

今後は，それぞれのキャッシュサーバの間のネットワークの帯域やルーティング情報，またオリジンサーバまでのネットワークの状態をもとに，ヒントサーバが適切と思われるキャッシュサーバを選択する機構を設計・実装し評価を行っていく予定である．

謝辞 本研究に対していろいろな助言をいただいたWIDEプロジェクトの諸氏，および奈良先端科学技術大学院大学の情報ネットワーク講座の諸氏に深く感謝いたします．

参考文献

1) Abrams, M., et al.: Caching proxies: Limitations and potentials, *Proc. 4th WWW Conference*, pp.119–133 (1995).
2) Berners-Lee, T., Fielding, R. and Nielsen, H.: Hypertext Transfer Protocol – HTTP/1.0, RFC1945 (1996).
3) The World Wide Web Consortium: CERN httpd (W3C httpd), http://www.w3.org/pub/WWW/Daemon/ (1996).

4) Chankhunthod, A., et al.: A Hierarchical Internet Object Cache, Technical Report 95-611, Computer Science Dep., Univ. of Southern California (1995).
5) National Laboratory for Applied Network Research: *Squid Internet Object Cache*, http://squid.nlanr.net/.
6) Wessels, D. and Claffy, K.: Internet Cache Protocol (ICP), version 2, RFC2186 (1997).
7) Netscape Inc.: Navigator Proxy Auto-Config File Format, http://www.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html (1996).
8) Berners-Lee, T., Masinter, L. and McCahill, M.: Uniform Resource Locators (URL), RFC1738 (1994).
9) SHARP Inc.: Super Proxy Script, http://naragw.sharp.co.jp/sps/index.html (1996).
10) Ross, K.W.: Hash Routing for Collections of Shared Web Caches, *IEEE Network*, Nov./Dec. 1997, pp.37–44 (1998).
11) Deering, S.: Host extensions for IP multicasting, RFC1112 (1989).
12) National Laboratory for Applied Network Research: Cache Registrations, http://ircache.nlanr.net/ (1996).
13) Feinler, E., Harrenstien, K. and Stahl, M.: NICNAME/WHOIS, RFC945 (1985).
14) 坂本岳史，井上博之，山口 英，尾家祐二：WWWキャッシュにおける隣接サーバ情報の設定支援機構，情報処理学会研究報告，マルチメディア通信と分散処理研究会，97-DPS-85, pp.201–206 (1997).
15) 知念賢一，井上博之，岡山聖彦，山口 英：WWWにおけるハイブリッド先読み代理サーバの設計と実装，信学会論文誌，Vol.J80-D-I, No.11, pp.907–915 (1997).
16) Hennessy, J.L. and Patterson, D.A.: *Computer Architecture A Quantitative Approach (2nd)*, pp.657–693, Morgan Kaufmann (1996).
17) Chaiken, D., Fields, C., Kurihara, K. and Agarwal, A.: Directory-Based Cache Coherence in Large-Scale Multiprocessors, *Computer*, Vol.23, No.6, pp.49–58 (1990).

(平成11年1月28日受付)
(平成11年9月 2 日採録)

井上 博之（正会員）

昭和40年生．昭和62年大阪大学工学部電子工学科卒業．平成1年同大学大学院前期課程修了．同年住友電気工業（株）入社．平成7年から10年にかけて奈良先端科学技術大学院大学情報科学研究科博士後期課程留学．分散処理，WWWキャッシュの研究に従事．電子情報通信学会，日本ソフトウェア学会各会員．

坂本 岳史

昭和47年生．平成8年大阪大学基礎工学部情報工学科卒業．平成10年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．同年松下電器産業（株）入社．

山口 英（正会員）

昭和39年生．昭和61年大阪大学基礎工学部情報工学科卒業．昭和63年同大学大学院前期課程修了．平成2年同後期課程を退学し，同大学情報処理教育センター助手に着任．平成4年奈良先端科学技術大学院大学情報科学センター助手，同助教授を経て，平成5年同大学情報科学研究科助教授．工学博士．インターネットアーキテクチャ，コンピュータセキュリティの研究に従事．

尾家 祐二（正会員）

昭和29年生．昭和53年京都大学工学部数理工学科卒業．昭和55年同大学大学院修士課程修了．同年日本電装（株）入社．昭和58年佐世保工業高等専門学校助手，昭和62年同助教授．平成2年九州工業大学情報工学部助教授．平成7年奈良先端科学技術大学院大学情報科学センター教授．平成9年九州工業大学情報工学部教授．平成9年8月から平成11年3月まで奈良先端科学技術大学院大学情報科学研究科教授（併任）．工学博士．インターネットのプロトコル，QoS制御方式，性能評価に関する研究に従事．IEEE会員．