NAIST-IS-DT9561004

博士論文

分散 WWW キャッシュシステムの自律分散化手法とキャッシュの適応制御に関する研究

井上 博之

2000年2月7日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

提出者：　井上 博之

審査委員：　山本 平一 教授
千原 國宏 教授
福田 晃 教授
山口 英 助教授

分散WWWキャッシュシステムの自律分散化手法とキャッシュの適応制御に関する研究*

井上 博之

内容梗概

インターネットの急速な広がりは World Wide Web（WWW）サービスによるところが大きい。その爆発的な普及にともない、現在ではWWWによるトラヒックがインターネット上の大半を占めるようになっている。そのトラヒックの伸びは年に200%とも言われ、ネットワークの輻輳やアクセス集中によるWWWサーバの過負荷、それらを原因とする利用者からみたWWWページの応答時間の増大が問題となっている。ネットワークのトラヒック削減と応答時間の短縮を目的として、WWWキャッシュが、またそれらを複数で連携して動作させる分散WWWキャッシュシステムが広く導入されてきている。現在の分散キャッシュシステムでは、キャッシュサーバ群の構成に関する設定は、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、他のキャッシュサーバあるいはキャッシュ間のネットワークの状態が変化するため、常に適切な構成を保つことは困難であるという問題がある。また、連係して動作するキャッシュ間のトラヒックを制御する仕組みがないため、細いリンクの帯域を圧迫してしまうという問題がある。この点は、コストが高くRTTが大きい静止衛星回線では特に顕著である。

まず最初に、インターネットのトラヒックとWWWキャッシュの役割、およびその問題点について検討する。ひきつづき、分散WWWキャッシュシステムについて解説し、これまでに提案されたキャッシュ管理手法、およびトラヒック制御方式について分析を行う。

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DT9561004, 2000年2月7日.

次に、キャッシュの自律分散化手法として、各キャッシュの内容と状態を把握するためのヒントサーバを導入した分散キャッシュシステムのモデルを提案し、設定・管理の自動化・簡略化を実現する方式について述べる。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わせに応答する。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、システムが自律分散動作していることを確認した。また、提案システムにおけるシステム全体のヒット率は、各キャッシュ毎の構成情報にキャッシュ間のトポロジーを反映させていないにもかかわらず、従来システムの理想的な設定でのヒット率と同一となり、提案システムの有効性を確認した。

最後に、トラヒック制御方式として、衛星ネットワークの特性を考慮したWWWキャッシュの適応制御方式の提案を行い、その実装および評価について述べる。静止衛星ネットワークはスター状のトポロジーを持ち、衛星回線のRTTが500msと大きいため、リム側のキャッシュで先読みを行うことが利用者からみた応答時間の短縮に効果がある。しかしながら、先読みは大きなトラヒックを生じ衛星回線の帯域がボトルネックとなるため特別な制御が必要となる。提案する方式は、衛星回線のトラヒック量をリアルタイムに監視し先読みを制限する方式と、利用者のアクセスパターンをもとに先読みすべきコンテンツを決定する方式である。さらに、ハブ側キャッシュは衛星回線帯域に余裕があればマルチキャストを使ってコンテンツをリム側に送付することでリム側でのヒット率の向上を図る。これらの方式を、アジアの複数の研究機関を衛星回線で結んだ研究ネットワークであるAI3ネットワークに適用し評価を行った。その結果、従来システムと比較してキャッシュのヒット率が15%向上し、オブジェクトの取得時間は約2割の短縮が実現でき、提案システムの有効性を確認した。

これらの分散キャッシュシステムの自律分散化手法とキャッシュの適応制御に関する研究によって、分散WWWキャッシュシステムの構成定義を維持管理する必要がなくなり、またボトルネックとなる回線の使用帯域とキャッシュ方式を制御することで、キャッシュのヒット率と応答時間を向上することが可能となった。

キーワード

分散WWWキャッシュ, ヒントサーバ, ICP, 衛星インターネット, マルチキャスト

Studies on Automomous and Adaptive Mechanisms for Distributed WWW Cache Systems*

Hiroyuki Inoue

Abstract

World Wide Web (WWW) technologies have resulted in the rapid growth of the Internet over the past several years, and WWW content has become the primary traffic on the Internet. WWW traffic has been expanding at a rate of 200% a year, bringing about the congestion of networks while hindering access to WWW servers. A distributed cache system for WWW infrastructure is introduced which reduces both network traffic and page retrieval latency. In this type of cache system, administrators are required to manually select which neighboring caches should be incorporated, based on their experience and expertise. However, it is difficult for administrators to update related management information in a dynamic and timely manner, in response to network or neighboring cache-server state changes. Furthermore, because of the lack of mechanisms to control the traffic along the narrow link between the cooperating cache servers, a problem will occur when traffic tends to flood the link, leading to bottlenecks. This phenomenon is particularly costly in a satellite link, which has a relatively narrow bandwidth and a larger round trip time.

First of all, the author showed the varying roles of WWW technology used with the Internet along with the related issue of WWW cache systems, providing descriptions of the applications and research objectives of the latter. Through the discussion, the author described the mechanisms involved in distributing WWW cache systems, while explaining cache management and traffic control methods proposed to date.

*Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9561004, February 7, 2000.

Secondly, the author proposed a new distributed cache architecture, which introduces a hint server that grasps the contents of each cache. From a query to a cache server, the hint server issues the order to the cache server indicating which cache server has the requested object. Through the deployment of such a cache system that includes the hint server, the author concludes that the proposed system achieves an autonomous automatically-managed configuration on each cache server, without degradation of the hit ratio in comparison with a conventional system.

Finally, the author proposed a WWW cache-control mechanism that adapts the traffic of satellite links and latency to retrieve a WWW object for the user. To prefetch a WWW object effectively reduces the latency of a satellite link; however, this technique also generates massive traffic and should be controlled. The author uses a control mechanism where the rim cache will decide whether to prefetch an object after observing the current traffic of satellite links in real time. The cache then decides which object should be prefetched, based on the users' access patterns. In addition, the hub cache of a satellite network distributes the frequently-accessed objects with multicast transmissions and improves the hit ratio on rim caches. The author adopted the mechanism for the AI3 satellite network, which connects several research organizations in Asia, and concludes that the proposed system improved the hit ratio of the cache 15% while reducing the mean retrieval time of an object 20%, compared to a conventional cache system.

Through the deployment of the proposed systems, the author successfully made the distributed cache system autonomous in its configuration, and one which achieves a higher hit ratio than the conventional system through the control of potential bottleneck traffic as well as the cache method.

Keywords:

distributed WWW cache, hint server, ICP, satellite internet, multicast

目 次

図目次

図目次

表目次

第 1 章　序論

この章では本論文の位置付けと意義を明らかにする。まず、近年のインターネットの発展とそのトラヒックの変化および WWW キャッシュシステムの意義について説明し、WWW キャッシュの問題点や研究課題について論じる。次に、本論文の主題である、分散 WWW キャッシュの自律的構成管理とキャッシュの制御方式について説明する。最後に、本論文の構成について述べる。

1.1.　インターネットとのトラヒックと WWW キャッシュ

インターネットは 20 世紀における第二の産業革命とも言われ、特にこの 5 年間で大きく発展を遂げてきた。18 世紀から 19 世紀にかけての産業革命では、町の小さな手工業に代わって産業機械を使用することで、低コストで大規模な生産が可能となりその後の社会構造を大きく変えることとなった。一方、インターネットにおいては、事務所や家庭などの環境で閉じたネットワークを形成していたコンピュータ同士が、インターネット技術すなわち実装に主眼をおいたアーキテクチャに依存しない共通のプロトコルを使用することで、相互に接続し情報を交換することが可能となった。そして、今まさに情報流通、物流、経済、起業、研究などの多くの分野に革命的な変革が起きつつある。

WWW（World Wide Web）が普及した理由として、「クリック」するだけで様々なマルチメディア情報を表示できるという情報入手の容易さだけでなく、自らが情報を作成し発信できるという魅力的な手段が提供されたことが挙げられる。HTML や JavaScript という比較的理解しやすい言語で WWW のページを

記述することで、動きのある画像や対話性のある画面を容易に作成し、世界中に情報を発信し、また情報共有を行うことができる。マルチメディア情報についても、WWWのページに埋め込まれた単なる画像や音声から、StreamWorksやRealAudioのようなストリーム型の音声・映像の配送、PointCastのようなプッシュ型のサービスなど多彩な形で提供されるようになってきた。また、インターネットを構成している複雑なネットワーク構成やプロトコルを意識することなく利用できる点もあって、広い社会層の人々に受け入れられている。

ビジネスへの展開も、当初は会社や製品の情報を一方的に公開するだけであったが、電子取引やCSCW（コンピュータ支援協調作業）の手段として、また消費者との対話の基盤として広く利用されるようになってきた。例えば、インターネット上の電子取引では店舗やショールームなどの設備を必要とせず、製造業者が直接消費者に商品販売を行うことも可能となる。このような直接販売方式の導入は、販売時の中間コストの削減をもたらすとともに、消費者の注文を受けてから製品を生産する受注生産（BTO: Built To Order）などを可能にした。また、個人利用者においても、公開された情報を入手するだけの使い方から、チャットのようなリアルタイムの情報共有やWWW技術を応用した知人とのメッセージ交換など利用形態が広がってきている。さらに、いわゆるイントラネットと呼ばれる会社などの組織内のネットワークにおいても、インターネット技術を応用した電子メール、ファイル共有、WWWベースの多くの応用ツール類が広く使用されるようになってきている。

WWWの歴史を振り返ってみる。1989年にCERN（European Laboratory for Particle Physics）のTim Berners-LeeによってWWWの基本的なHTML（Hyper Text Markup Language）およびHTTP（HyperText Transfer Protocol）という仕組みが設計され、1991年8月にCERNからリリースされた。この時点でのWWWクライアント（ブラウザ）はテキストベースのものであった。1993年にイリノイ大学のNCSA（National Center for Supercomputing Applications）で開発され配布されたMosaicという、ウインドウシステム上で動作するWWWブラウザによって、広くWWWの利便性が認識されるようになった。この時点では世界中のWWWサーバの数は200程度にすぎなかった。1994年には、第1回のWWW ConferenceやW3C（World Wide Web Consortium）のミーティングが

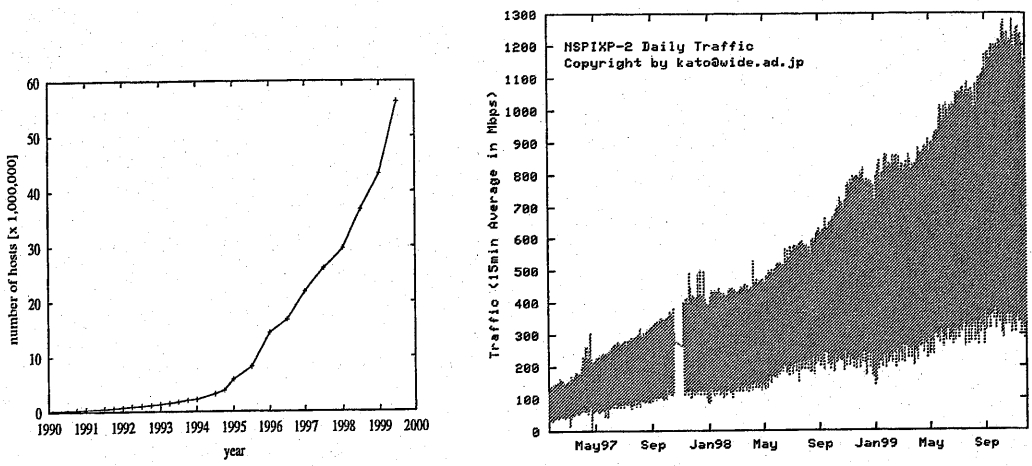
表1.1 インターネット利用数の統計

日本のインターネット人口（1999年11月）	約1,540万人
同、世帯普及率（1998年度）	11％
同、企業普及率（同）	80％
日本のWWWサーバ数（1999年2月）	7.5万台
同、コンテンツ総量（同）	約1TByte
jpドメイン登録数（1999年10月）	10万
世界のインターネット人口（1999年2月）	1.5億人
世界のWWWサーバ数（1999年7月）	5600万台
世界のドメイン登録総数（1999年7月）	170万

行われている。1994年10月にNetscape社によるWWWブラウザの公開が行われ、またその頃から米国や日本で個人向けの商用のインターネットサービスがはじまった。[1][2][3]

また、WWW技術はその拡張のための副産物として別の優れた技術を生み出していることも忘れてはならない。WWWを使った安全な電子取引のための通信暗号化技術としてSSL（Secure Socket Layer）がNetscape社によって開発され、現在は標準的なHTTP通信の暗号化手段としてWWW上のプライバシ保護技術として利用されている。別の目的でSun Microsystems社によって開発されたJava言語は、WWWブラウザでサポートされたこともあって現在は標準的なブラウザの動作記述言語として広く使用され、WWWブラウザを使用して対話型のページやさまざまなアプリケーションが実行できるようになった。RealAudioはWWWと連動してストリームで音声を配送する仕組みを確立し、簡単な操作で音声や映像をリアルタイムで再生できるようになった。これらの技術が現れた1995年春頃からインターネットの主要なトラヒックはWWWに代わり、現在のような爆発的な利用につながっている。

最近のインターネットの普及を統計的な数値でみると表1.1のようになる[4][5]。個人向けのインターネットの商業利用が開始されたのが1994年頃であるから、わずか5年で世帯普及率が10%を超えたことになる。パーソナルコンピュータや携帯電話が世帯普及率10%を超えるにはいずれも十数年を要したことを考えても急



(a) Number of hosts in the Internet　　(b) Traffic at NSPIXP2

図 1.1　インターネットのホスト数と NSPIXP2 のトラヒックの変化
（(a) は文献 [6]、(b) は文献 [7] より引用）

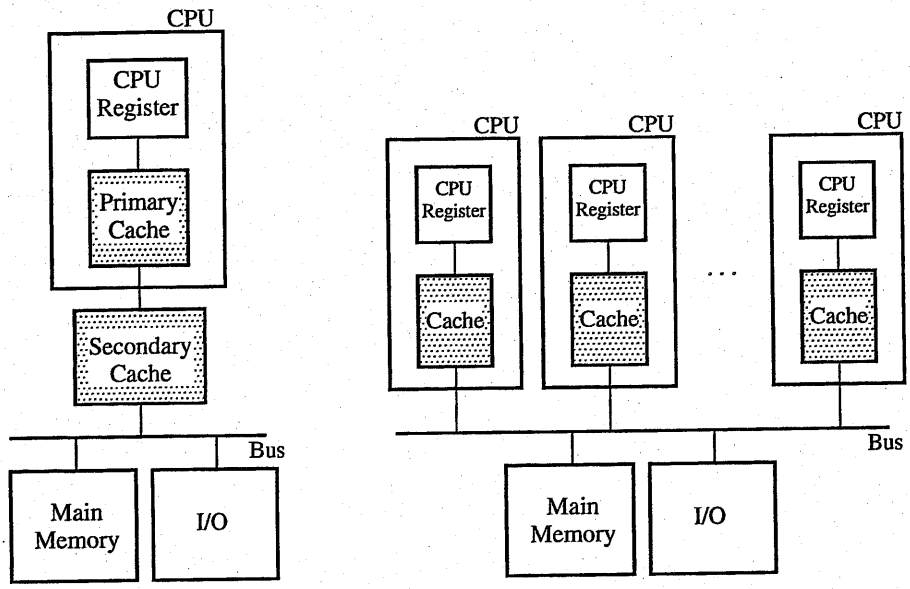
激な普及がわかる。図 1.1(a) にインターネットのホスト数の増加をグラフにしたものを示す。1 年間で約 1.5 倍に増加しており、過去 4 年間で 7 倍にもなっている。この傾向は今後も続くと考えられる。インターネット全体のトラヒックを正確に計測することはできないが、日本の代表的なインターネット相互接続点（IX: Internet Exchange）である NXPIXP2 で計測したトラヒックの変化を図 1.1(b) に示す。1 年間で IX を通過するトラヒックは 2 倍ずつになっており、この傾向もしばらく続くと考えられる。[6][7]

このようにインターネットにおける WWW の利用は指数関数的に伸びており、その爆発的増加はアクセスの増加とそれに伴なうネットワークの混雑、また情報の取得時間の増大という問題を引き起こしている。数年前から、インターネット上のトラヒックを減少させるために、WWW キャッシュという技術が提案され広く適用されてきている [8]。WWW キャッシュは、HTTP の要求を中継するためのプロキシサーバにおいて、中継したデータを一時的にメモリやハードディスクに格納する機能を持たせ、次に同じ要求が来た際にそれを返す形で実装された。なお、同一の要求であることは、オブジェクトの所在を表す URL（Uniform Resource Locators）[9] の同一性で識別する。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（HTTPサーバ）[10]にキャッシュ機能を持たせたものである。キャッシュサーバはWWWクライアントからみてプロキシサーバ（代理サーバ）として扱われ、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基本的に変わっていない。

その後、Harvestプロジェクト[11]において、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一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もっとも広くインターネット上で使用されているSquidキャッシュ[12]である。また、研究ベースですすめられてきたWWWキャッシュシステムの成功をみて、プロキシサーバおよびルータやブリッジのような形態でネットワークに設置できる専用の装置がいくつか商品化され入手できるようになっている[13][14]。WWWキャッシュの種類やその機能およびプロトコルについては、2章で詳しく述べる。

元来、キャッシュ技術は、1960年代にコンピュータシステムにおいてCPUとメインメモリのサイクルタイムの差を吸収し、性能を向上させるために導入された[15]。CPUのキャッシュは、単一プロセッサやマルチプロセッサ、階層型や分散型などでいくつかの分類を行うことができる。代表的な例を図1.2に示す。(a)は単一プロセッサにおける階層型キャッシュで、CPUが1個の一般的なパーソナルコンピュータになどがこの構成をとる。(b)は密結合の対称型マルチプロセッサシステム（SMP）で、CPUを複数持つパーソナルコンピュータやワークステーションなどがこの構成を持つ。他にも疎結合のマルチプロセッサシステムなどがあるが、一般に、CPUのキャッシュは別々に置かれその整合性をとる必要はないため、ここでは議論しない。CPUのハードウェアキャッシュでは、キャッシュの中身はCPUのプログラムやデータであり、キャッシュの一貫性（cache consistency）が保てないとシステムが正常に動作しないため、その応答時間と一貫性に厳密な保証が要求される。しかしながら、WWWキャッシュでは、WWWサーバや他のキャッシュサーバからの応答時間や内容の一貫性は一般に保証できていない。WWWキャッシュにおいて一貫性が保てない場合は、クライアントが実際のWWWサーバが返すべきオブジェクトとは異なる古いオブジェクトを返すことになる。この場合でも、CPUキャッシュのようにシステム全体が動作しなくなるということは起こ



(a) Hierarchical Cache for Uniprocessor System

(b) Cache for Symmetrical Multiprocessor System

図 1.2 コンピュータシステムの構成と CPU のキャッシュ

らないとはいえ、利用者が気がつかないうちに古い情報を得てしまうという問題がある。これは、もともとの HTTP のプロトコルに情報提供側の変更を通知する仕組みはないことに起因するが、WWW のリンクが片方向であることが WWW の情報提供者の負担の軽減およびシステムの単純化に寄与しており、一概に問題であるとは言えない。また、インターネットのような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のキャッシュと同様の議論は難しく、WWW キャッシュのためのシステムモデルを確立する必要がある。

1.2. WWWキャッシュシステムとその問題点

WWWキャッシュサーバを使ったWWWアクセスシステムの典型的な構成を図1.3に示す。WWWクライアント（ブラウザ）からのHTTP要求はキャッシュサーバに送られ、キャッシュサーバが別のHTTP要求をWWWサーバに対して行い、アクセスを代行する。キャッシュサーバは、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している。なお、このような形態のキャッシュサーバは、クライアントからみるとHTTPのプロキシサーバとして振舞うことになる。

また、2章で詳しく述べるように、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（以下、隣接キャッシュサーバ）にそのオブジェクトの有無を問い合わせる。オブジェクトを保持している隣接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。あるキャッシュサーバから他のキャッシュサーバの内容を知るために、ICP（Internet Cache Protocol）[16]というUDPを使った問合せ･応答型のプロトコルが使用されている。

現在、研究されているWWWキャッシュシステムの問題点や研究課題をまとめると以下のようになる。[17][18]

1. キャッシュシステムの構成管理

   複数のキャッシュサーバからなる分散キャッシュシステムにおいては、クライアントと各キャッシュサーバの管理者は、ネットワークの状態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

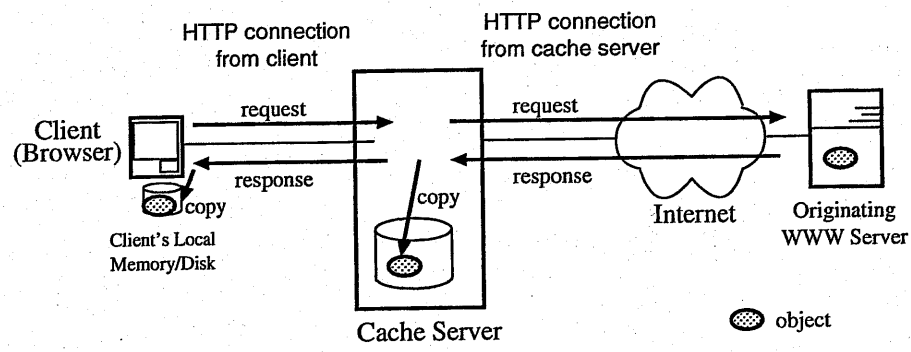


図 1.3 WWW キャッシュシステムの基本的な構成

し、それぞれのキャッシュサーバに対して手動で静的に設定を行う必要がある。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、管理者に常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。

クライアントの構成情報については、WPAD というキャッシュ資源の探索方式が提案されており [19]、クライアントのプロキシおよびキャッシュに関連する設定は自動化できる。なお、透過型のキャッシュサーバを使用すれば、クライアントは明示的にキャッシュサーバをアクセスする必要がなく、クライアントからみた構成管理の問題は発生しない。次に、分散キャッシュシステムの複数のキャッシュサーバの構成情報については、ICP の問合せに IP マルチキャストを利用する方式や、キャッシュサーバが自身の構成情報を特定のデータベースに登録し他のキャッシュサーバがそれを参照する方式 [20] などがある。これらの方式を使用しても、入手できるキャッシュサーバ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必要があり、またキャッシュ内容に基づくキャッシュサーバの選択ができないという問題点がある。

また、他のキャッシュサーバを選択する際に、本来の WWW サーバあるいは他のキャッシュサーバに至るネットワークの状態をもとに、アクセス先の選択や取得したオブジェクトをキャッシュすべきかどうかという判断を適切

に行う機構が十分ではないという問題もある。

2. 一度しかアクセスされないオブジェクト

キャッシュサーバが取得しキャッシュに格納したオブジェクトのうち、キャッシュヒットしない、すなわち一度しかアクセスされないものの割合は約5割にもなる。この割合を減少させることで、キャッシュの利用効率を改善することができる。利用者のアクセスの状況を統計的に分析することで、再利用の可能性を推定し、ヒット率を向上させる研究などが行われている [21]。

3. キャッシュ領域のサイズ

キャッシュ領域のサイズを大きくしていくと、キャッシュのヒット率は向上するが、その関係は log であり、キャッシュのサイズを倍にしても、ヒット率の劇的な向上が見込めるわけではない。WWW キャッシュのヒット率はおおよそ 30 ～ 50%と言われ、筆者の運用結果でも同様の値が得られている。また、キャッシュのサイズを無闇に大きくすると、オブジェクトを検索する時間が増大し、かえってキャッシュサーバ自身の応答時間の悪化を招くという副作用も生じてしまう。

4. 隣接キャッシュのヒット率

分散キャッシュシステムにおける、隣接キャッシュサーバのヒット率の寄与割合はせいぜい 5 ～ 15%程度であることが運用からわかっている。このような低いヒット率のために、複雑なキャッシュシステムを構成する必要があるのかという議論がある。

5. キャッシュの一貫性

キャッシュに置かれたオブジェクトが古くなり、本来の WWW サーバ上のオブジェクトとの不整合が生じる可能性があるが、既存の HTTP の仕組みでは WWW サーバ側で変更されたことを通知するような仕組みはない。よって、何らかのキャッシュに保存されたオブジェクトと本来のサーバのオブジェクトの一貫性を保つ方策が必要となる。オブジェクトがヒットした際に、本来のサーバにそのオブジェクトが更新されているかどうか必ず問い合わせする方法もあるが、キャッシュ本来の目的であるトラヒックと応答時

間の削減という効果が損なわれてしまう。よって、キャッシュの更新アルゴリズムとして、オブジェクトの最終更新時刻、キャッシュへの取得時刻、利用者からの最終アクセス時刻、などを比較して、キャッシュから破棄するかどうか決定するなどのヒューリスティックな手法が研究されている。

6. コンテント・ネゴシエーションされたオブジェクトの扱い

WWWクライアントからの要求として、ブラウザの使用言語や取扱い可能な画像形式を指定することで、それに応じた内容をWWWサーバが応答する、コンテント・ネゴシエーションという仕組みがある。また、ブラウザのバージョンや製品に応じて、異なるページを応答するようなWWWサーバも存在する。このような、クライアントやその設定によって内容が変化するオブジェクトは単純にキャッシュするわけにはいかない。現在のキャッシュシステムでは、ネゴシエーションされたかどうかを検出する仕組みはなく、他のオブジェクト同様にキャッシュしてしまうため、結果としてWWWサーバが意図した内容と異なるオブジェクトを利用者が得てしまうことがある。この問題を回避するためには、クライアントからの要求内容とその応答を全てキャッシュすることで、要求が同一であることを判定することは可能である。しかし、実現には、要求内容を記憶する領域の増大、要求が一致するキャッシュ内容を検索する手数の増大、オブジェクトの同一性がURLだけで判断できなくなってしまうためにキャッシュシステム全体に影響が及ぶ、などの課題を解決する必要がある。

7. 文化的に異なる組織間でのキャッシュ

異なる国家間など使用言語や文化的背景が異なる組織同士で、キャッシュを協調動作させた場合、キャッシュされるコンテンツの傾向が異なるため、相互のキャッシュ間のヒット率が低くなってしまうという問題がある。例えば、日本と中国の組織で相互運用を行った場合を考える。日本のキャッシュサーバからは、日本語で記述されているオブジェクトは、中国のキャッシュへ問合わせを行わず、英語や中国語で記述されているオブジェクトは問合わせの対象とする、などの手法を用いることでキャッシュの利用率と応答時間を改善できる可能性がある。

8. キャッシュできないコンテンツの増加

商取引や検索エンジンのような問合わせとそれに応じた結果を得るようなコンテンツ、またチャットに代表される対話的なコンテンツにおいては、同じ URL でも毎回応答が異なる。このような応答にはキャッシュされないように、HTTP の応答ヘッダあるいは HTML 中に「キャッシュするな」という指令が埋め込まれている。この種のコンテンツの増加によって、結果的にキャッシュのヒット率が低下してしまうという問題がある。しかしながら、本来キャッシュしてはならないものであり、そのトラヒックを減少させることは困難である。よって、キャッシュのヒット率を議論する場合、キャッシュしないことを明示的に指示されているオブジェクトは別に扱うべきである。

9. キャッシュの不正使用

プロキシキャッシュサーバを経由することで、WWW サーバからはアクセス元のクライアントの IP アドレスなどを隠蔽することができる。これを悪用して、他者に迷惑をかける行為を行ったり、不正なアクセスのための踏台として利用されることが社会的な問題となってきている。そのため、誰もが利用できるようなキャッシュサーバは減少する傾向にあるが、このような消極的な解決策ではなく、キャッシュサーバの利用者を認証する仕組みや、利用者を追跡する仕組みを実現していく必要がある。

10. アクセス記録のプライバシ

キャッシュサーバには、一般に、利用者の IP アドレスやアクセスしたオブジェクトの URL、アクセス時刻などがログとして残る。多くの検索エンジンでは、URL には利用者が問い合わせした文字列がそのまま含まれている。このように、キャッシュのログには、利用者のプライバシに関する情報が含まれており、その管理・運用には十分な注意が必要である。

11. コンテンツ提供者の論理

WWW サーバのコンテンツ提供者からみると、コンテンツはできるだけ多くの利用者によって参照されることを欲し、また実際に参照した利用者を正しく把握できることを望んでいる。WWW キャッシュやプロキシを経由

すると、それらの情報を正確に得られなくなるため、コンテンツ提供者にとってはやっかいな問題となる。これを回避するために、コンテンツを強制的にキャッシュしないように提供者側で設定する場合もある。ネットワークのトラヒックの減少と排反する問題であり解決することは困難である。

12. キャッシュの著作権

WWW サーバが提供するコンテンツをキャッシュすることは、デジタル情報のコピーであり、著作権侵害であるという議論がある。米国では、キャッシュがコピーに当たるか否かなどの訴訟が実際に起こされており [22]、1998 年に成立したデジタルミレニアム著作権法 [23] では、キャッシュに関するプロバイダの責任限定などが明確に定義された。日本では、キャッシュ内容は著作権法上の私的利用の範囲と解釈されるかどうかも含め、WWW キャッシュの著作上の問題に関する議論ははじまったばかりである [24]。

13. ストリーム型メディアの増加とキャッシュ

音声 (音楽) や映像を、クライアントで取得し蓄積してから再生を行うのではなく、受信しながら同時に再生を行うタイプの情報提供サーバが増加している。代表的な例として、Real Networks 社の Real[25]、Cisco 社の IP/TV[26] などがある。このような情報提供形態をストリーム型の配送と呼び、そのコンテンツをキャッシュする方式が研究されている。また、MP3 （MPEG-1 Audio Layer 3）や MPEG-4 などに代表される、低ビットレートの符号化方式を使用したコンテンツも増加してきている。これらのコンテンツは、従来の WWW キャッシュの対象としているオブジェクトと比較して、数メガバイトから数十メガバイトとサイズが非常に大きくなることから、効率的なキャッシュ方式を確立していく必要がある。例えば、Akamai 社では、ネットワークの状態に応じて最適なサーバを選択する機構を、ストリーム型メディアなどの大規模コンテンツに適用したシステムを開発している [27]。

14. キャッシュサーバの負荷

キャッシュサーバ、特にプロキシの形態をとるサーバには全ての WWW アクセスのトラヒックが集中するため、その負荷が高くなってしまうことや、

故障時には全てのクライアントからのアクセスができなくなってしまうという問題がある。負荷分散と耐故障の手法として、DNS（Domain Name System）を使ったラウンドロビン [28]、ハッシュルーティング [29][30]、4層スイッチの利用、などがある。これらの手法は WWW サーバの負荷分散にも同様に使用されており、クラスタサーバと呼ばれる。

キャッシュサーバの装置の構成に着目すると、そのキャッシュ内容を記憶する媒体としてメモリとハードディスクの 2 階層からなり、メモリのサイズは数十から数百メガバイト、ハードディスクのサイズはその十倍程度である。ハードディスクのアクセス速度はメモリと比較して非常に低速であるため、その入出力がボトルネックにならないような設計を行う必要がある。ディスクのストライピング（RAID 0）は有効な手法であるが、CPU バスなどを含めた性能設計が必要である。キャッシュに使用するハードディスク上に、WWW オブジェクトの特性を考慮した高速なアクセスができる専用のファイルシステムを搭載している商用製品もある。また、ネットワークの自体の速度が 2.5 Gbit/s や 6 Gbit/s のように高速になるにつれて、1 台のキャッシュ装置で処理可能な速度との乖離が起こり、1 つのオブジェクトのキャッシュと転送を複数の装置で分散処理する必要がでてくるであろう。

15. キャッシュシステムの評価基準の確立

WWW サーバ用の性能を検査するツールとして、SPECweb[31] というベンチマークプログラムが 1996 年に開発された。一方、WWW キャッシュ専用の性能測定プログラムとして、IRCache Project によって Web Polygraph [32] というツールが 1998 年に開発された。このツールを用いてキャッシュ装置のベンチマークを競うイベントが定期的に行われ [33]、キャッシュ開発者らの情報交換の場としても重要となっている。また、キャッシュ製品ベンダによる、Inktomi 社の Large Scale Benchmark、CacheFlow 社の Performance Tool、Network Appliance 社の NetCache Load Generator などのツールがある。これらの性能評価ツールでは、利用者のアクセスのパターン、コンテンツの分布、ネットワークの状態の変動などを、いかに実際のネットワークの挙動に近似できるかが課題となっている。

1.3.　本論文で扱う題目

前節で議論したWWWキャッシュの問題点のうち、本論文では、1.で挙げた分散キャッシュの自律分散化の問題と、2., 4., 5. で挙げたキャッシュのヒット率を低下させる問題を解決できるようなキャッシュの制御方式に関して検討を行う。

1.3.1　キャッシュサーバの自律分散化方式

分散キャッシュシステムにおいて、構成定義情報が存在しない状態で自律的にキャッシュサーバを選択する方式について検討および実装・評価を行う。システムを構成する各キャッシュサーバの状態とその内容を把握するディレクトリ参照型のサーバ（ヒントサーバ）を導入することで、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。

1.3.2　衛星ネットワークにおけるキャッシュ制御方式

キャッシュサーバからWWWサーバや他のキャッシュサーバに至る経路やネットワークの状態を考慮してオブジェクトの取得先の選択を行う方式や、利用者の要求を統計的に分析してオブジェクトを先読みしてキャッシュのヒット率と応答時間を改善する方式について検討する。静止衛星を使ったネットワークでは、その衛星リンクのコストが高いため、リンクの帯域を有効に使用することが重要であることから、衛星ネットワークの特性を考慮したWWWキャッシュ制御方式の提案を行う。スター状のトポロジーで構成されている衛星ネットワークにおいては、リムに位置する国からのアクセスは静止衛星回線を経由するため回線自体の

RTTである約500msが大きな遅延の原因となる。帯域幅の小さい衛星回線がボトルネックになることから、単純にリム側にWWWキャッシュを設置することである程度の改善は可能であるが、実際には衛星回線のトラヒックの制御と衛星回線のRTTに起因する応答時間の減少を如何に行うかが問題となる。提案システムの適用として、AI3というアジアの複数の研究機関を衛星リンクで結んだ研究ネットワークにおけるWWWキャッシュの制御方式を検討し実装・評価する。

1.4. 本論文の構成

本章に引続き、第2章では、分散WWWキャッシュシステムの特徴および問題点について解説し、これまでに提案されたキャッシュ管理手法およびトラヒック制御方式について述べる。第3章では、各キャッシュの内容と状態を把握するためのヒントサーバを導入した分散キャッシュシステムのモデルを提案し、設定・管理の自動化・簡略化を実現する方式について述べる。第4章では、衛星ネットワークの特性を考慮したWWWキャッシュ制御の方式の提案と、AI3ネットワークというアジアの複数の研究機関を衛星回線で結んだ研究ネットワークに適用し、実装および評価を行った結果について述べる。第5章は結論で、本研究の総括を行い、今後の課題および予定について述べる。

第 2 章　分散キャッシュシステム

分散WWWキャッシュシステムを構成する基本技術として、WWWキャッシュの仕組み、種類、プロトコルなどについて説明し、その特徴や問題点について述べる。

2.1.　WWWキャッシュ技術

WWWキャッシュはオブジェクトの再利用とネットワークのトラヒックを減少させるために導入され、当初はプロキシサーバの形態で実装された。一般にインターネットにアクセスする場合のボトルネックになるのは、ローカルネットワークとインターネットとの間の経路であることから、キャッシュサーバはそのローカル側に設置されたファイアウォールを兼ねるプロキシサーバとして設置されることが多かった。その後、複数のキャッシュサーバを協調動作させる分散キャッシュや、プロキシ型とは異なるタイプのキャッシュが登場している。

本節では、ファイアウォールとプロキシサーバについて説明し、WWWキャッシュの技術とその問題点などについて述べる。

2.1.1　ファイアウォールとプロキシサーバ

大学や企業などのネットワークにおいては、その組織内のネットワークとインターネットとの間に、セキュリティ上の防火壁であるファイアウォール [34][35] を設置していることが多い。ファイアウォールの種類によっては、組織内のネットワークにあるコンピュータからインターネット上のWWWサーバへ直接アクセスできないことがある。この問題を解決するために、WWWサーバへのアク

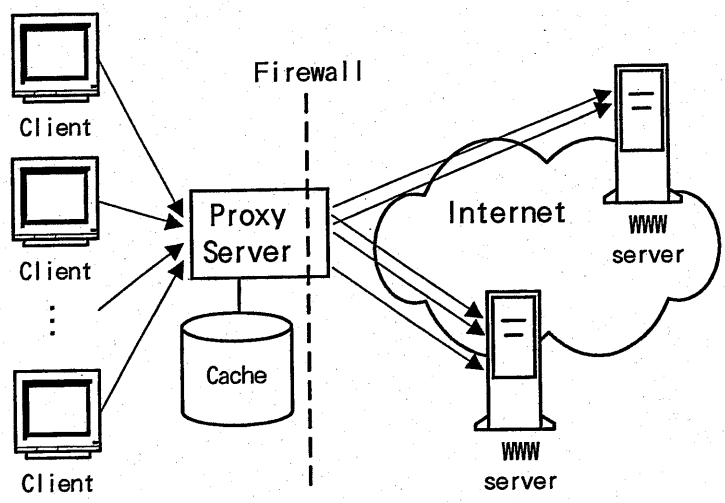


図 2.1　ファイアウォールとプロキシサーバ

セスを代行するプロキシサーバ (proxy server、代理サーバ) をインターネットとの境界部分に設置する。WWW クライアント（ブラウザ）はアクセス要求をプロキシサーバに送り結果を受け取ることにより、結果としてアプリケーションからはファイアウォールを意識しない透過的なアクセスを実現することができる。このような形態のファイアウォールをアプリケーションゲートウェイ型と呼ぶ。

プロキシサーバを使った典型的な構成を図 2.1に示す。WWW クライアントからの要求はプロキシサーバに送られ、プロキシサーバが本来の WWW サーバへのアクセスを代行する。プロキシサーバには複数のクライアントからのアクセス要求が集まること、またアクセスを代行しクライアントに中継することから、アクセスした WWW オブジェクトを一旦保存し、次回同じアクセスがあった時にそのオブジェクトを応答するというキャッシュ動作を持たせるアイデアが出てきた。

プロキシサーバで提供することが可能な機能をまとめると次のようになる。

- プロキシサーバで、一度アクセスされた WWW オブジェクトのキャッシュを行うことにより、以降に同じデータをアクセスした同じまたは異なるクライアントに対して、キャッシュに保持していたオブジェクトを返すことができる。本来の WWW サーバにアクセスする必要がなくなるため、ネット

ワークのトラヒック及び WWW サーバの負荷の軽減、クライアントから見た応答時間の短縮が期待できる。キャッシュ機能を持たせたプロキシサーバのことを、特にキャッシングプロキシサーバ (caching proxy server) または単にキャッシュサーバと呼ぶ。

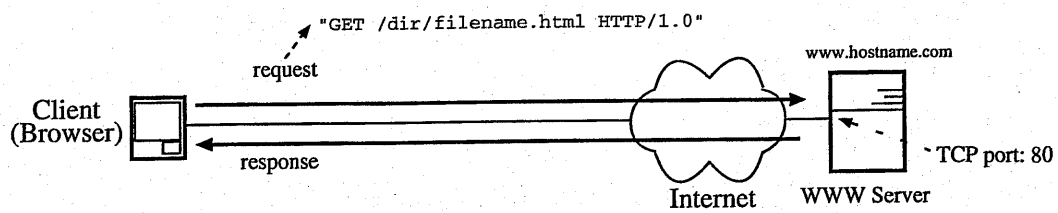
- 本来の WWW サーバに直接アクセスを行うのはプロキシサーバであり、クライアントの IP アドレスに代表される組織内のネットワーク構造などを隠蔽することができる。

- WWW オブジェクトがプロキシサーバを経由する際に、データの加工や調査を行うことができる。例えば、オブジェクトの日本語漢字コードや画像形式をローカルな形式に正規化することや、アクセスの統計情報を得ることに利用できる。例えば、日本で開発されたプロキシサーバとして、1994 年から電子技術総合研究所のメンバによって開発された DeleGate[36] がある。DeleGate は日本語や中国語の漢字コード変換や NetNews, CU-SeeMe などの WWW 以外のプロキシ機能を実現し独自の発展を遂げてきている。

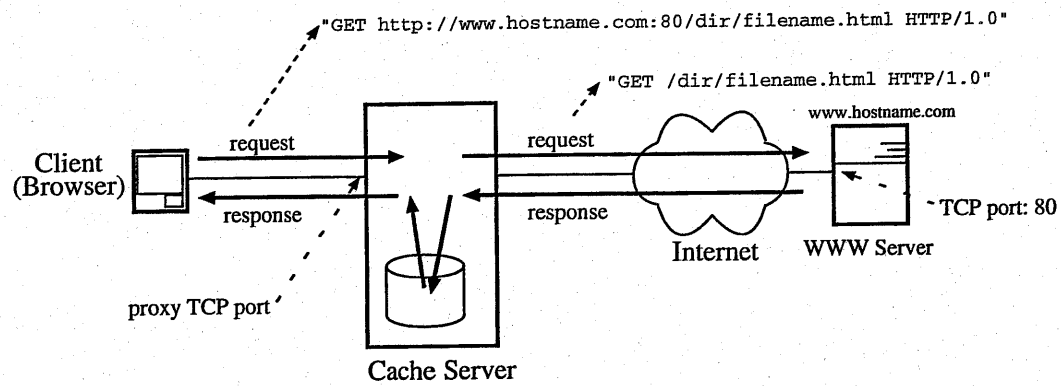
2.1.2 クライアントとプロキシサーバ間のプロトコル

WWW サーバ (HTTP サーバ) に、クライアントが直接アクセスするときと、プロキシサーバ経由でアクセスするときのプロトコルはいずれも HTTP[37] [38] を使用するが、その違いについて説明する。直接アクセスするとき、クライアントは URL で指定された WWW サーバの指定された TCP ポートに対して接続を行い、WWW サーバにファイルパスの部分だけを送る。一方、プロキシサーバに対してはクライアントは完全な URL をそのまま送る。プロキシサーバは URL を解釈し、URL に記述されたプロトコルで指定されたサーバに対してアクセスを代行し、得られた結果をクライアントに中継する。

WWW クライアントが `http://www.hostname.com:80/dir/filename.html` という URL にアクセスしようとした場合を考える。プロキシサーバを使用せず、直接 WWW サーバにアクセスするときは、www.hostname.com というホストの TCP 80 番ポートに対して HTTP コネクションを行い、"`GET /dir/filename.html HTTP/1.0`" という要求を送る。一方、プロキシサーバを経由してアクセスするとき



(a) WWW client accesses WWW server directly



(b) WWW client accesses WWW server via proxy server

図 2.2 プロキシサーバを使用する場合のプロトコル

は、WWW サーバのホスト名を解決することなく、プロキシサーバに対して HTTP コネクションを行い、"GET http://www.hostname.com:80/dir/filename.html HTTP/1.0" という要求を送る。そして、プロキシサーバはこの要求を解析し、WWW サーバ www.hostname.com の TCP 80 番ポートに対して HTTP コネクションを行い、"GET /dir/filename.html HTTP/1.0" という要求を送り、その応答をクライアントに中継する。

HTTP/1.0 では 1 つのコネクションで 1 つのオブジェクトを取得するようになっているが、コネクションを新しく設定することはオペレーティングシステム (OS) やネットワークに大きな負担をかける処理であり、同じ WWW サーバあるいはキャッシュサーバに対して何度もコネクションを設定することは、OS およびネットワーク資源の利用効率を低下させる。HTTP/1.1[38] で導入された Persistent

Connection と Pipelining はこの問題を解決することを意図している。Persistent Connection は同じ WWW サーバに対してコネクションを維持したまま、次のオブジェクトの要求を出すための拡張で、また Pipelining はオブジェクトを完全に取得する前に次の要求を出すことを可能とする拡張である。これらの機能の効果を調べた実験結果によると、ネットワーク上を流れた総パケット数が数分の一になり、必要とした OS の資源（ここでは socket の数）が大きく減少する効果が確認されている [39]。

2.1.3 プロキシサーバによるキャッシング

1994 年に CERN で開発された WWW サーバは、HTTP サーバであると同時に、WWW のプロキシ機能も持っている [10][40]。CERN の WWW サーバは 1995 年頃まではキャッシューバとして広く使われていた。しかし、CERN のサーバは計算機に与える負荷が非常に大きく、WWW のトラヒックが増大するに従って満足な性能が得られなくなってきた。CERN のサーバは Unix 上で、HTTP の TCP コネクション毎にプログラムのプロセスの複製を作成する実装になっている。すなわちプロキシサーバの同時接続数だけプロセスが生成されることになり、現実的な毎秒数百から数千の処理を行うことは不可能であった。なお、現在最も広く使用されているキャッシュサーバは後述する Squid[12] キャッシュである。Squid の実装では、HTTP のコネクションとキャッシュ管理を行う処理を単一のプロセスとし非同期 I/O を行うことで解決している。なお、Squid では、ホスト名から IP アドレスへの解決処理と、FTP サーバへのアクセス処理に関しては別プロセスとして実装されている。この理由として、これらの 2 つの処理は状態遷移が複雑であるため、別プロセスとして実装するほうが有利であったことが挙げられる。

なお、キャッシュサーバを設置する場所としては、クライアントと同じローカルネットワークあるいは広帯域の回線で結ばれたネットワークに設置されるべきである。なぜならば、キャッシュサーバは応答時間の短縮と広域網におけるトラヒック量の削減を実現するために設置されるからである。よって、ネットワークに起因する遅延を考える場合、キャッシュサーバがアクセスする WWW サーバあるいは他のキャッシュサーバまでの遅延時間とそのばらつき（分散）が重要な要素となる。

2.1.4　WWWキャッシュの性能

WWWキャッシュの性能指標としては以下のものがある。なかでも、キャッシュのヒット率とクライアントからみた応答時間は、しばしばキャッシュ性能の代表的な指標として使用される。

ヒット率
　アクセス要求の数の総数のうちキャッシュにヒットした（オブジェクトが存在した）アクセス数の割合。

応答時間
　クライアントがキャッシュサーバにアクセスしてから、オブジェクトを取得完了するまでの時間。

スループット
　単位時間あたり（通常1秒あたり）に処理した要求数。

最大同時処理要求数
　キャッシュサーバで同時に処理可能な要求数。

トラヒック減少割合
　キャッシュを経由することで減少したパケット数およびバイト数の割合。

また、キャッシュのヒット率としては、アクセス要求の数の総数のうちキャッシュにヒットしたアクセス数の割合で表す“アクセス数ベースのヒット率”と、アクセスの結果取得したオブジェクトの総バイト数のうちキャッシュにヒットしたオブジェクトのバイト数で表す“バイト数ベースのヒット率”の2通りが考えられる。一般にキャッシュのヒット率と言う場合は、前者を指すが、トラヒックやネットワーク負荷を考慮する場合は後者の指標を使用したほうが便利な場合もある。

なお、1.2節で述べたように、ヒット率を下げる要因として、WWWのチャットのようなクライアントプル型のアプリケーションや、PointCastのようなクライアントプッシュ型のアプリケーションが挙げられる。

2.2. WWW キャッシュの分類

WWW キャッシュには大きく分けて、プロキシ型と透過型の 2 種類の方式がある。プロキシ型ではこれまで説明してきたように、HTTP 等の TCP セッションをプロキシサーバで一旦終端し、WWW サーバからはプロキシサーバがアクセスするようにみえる。一方、透過型では、TCP セッションを途中のルータあるいはブリッジ装置が横取りし、クライアントからは WWW サーバと通信しているようにみえ、WWW サーバからはクライアントが直接アクセスしているようにみえる。すなわち、後者の方式ではキャッシュサーバの存在は検知できず透過的である。

また、アクティブキャッシュ技術として、クライアントの要求とそのコンテンツの内容をもとに、次に要求されるであろうコンテンツを予測しキャッシュに先読みしておく先読みキャッシュサーバや、サーバ間でコンテンツの複製を行うレプリケーションや、あるキャッシュサーバから別のキャッシュサーバにコンテンツを送り込む配送について、以下の節で説明を行う。

2.2.1 プロキシ型キャッシュ

プロキシサーバは WWW クライアントからの HTTP セッションを終端し、アクセスしたい WWW サーバあるいは別のプロキシサーバに対して、新しい TCP セッションを形成し、2 つのセッションの間でデータを中継する。よって、プロキシ型のキャッシュはアプリケーションゲートウェイ型のファイアウォールとの親和性がよく、インターネットに対する匿名性にも優れている。また、WWW サーバとクライアントの間がルーティングされていない場合にも有効である。しかしながら、多くの TCP セッションを処理する必要があることから負荷が高くなりがちである [41]。

プロキシ型キャッシュにはいくつかの欠点がある。前述のように、プロキシ設定は個々のクライアントに入力されなければならず、ネットワーク管理にとっては負担となる。またプロキシサーバは、インターネットとの接続点に通常設置されるため、ボトルネックかつ障害に対して脆弱な構成になりがちであり注意が必要である。

2.2.2　透過型キャッシュ

透過型キャッシュはルーターまたはブリッジ（スイッチ）装置として動作し、WWW アクセスするための HTTP 等の TCP コネクションを判別し、自動的にキャッシュサーバに転送する。よって、利用者はクライアントのプロキシ設定を行うことなく、キャッシュサーバの利用ができる。

透過型キャッシュは、ネットワーク構成という面からみるとプロキシ型キャッシュほど複雑性はないが、TCP セッション自体を操作しキャッシュサーバが WWW サーバになりすます必要があるため実装はより複雑となる。透過型キャッシュの基本的な構成を図 2.3に示す。ルータあるいはブリッジ装置はクライアントと WWW サーバの間に設置され、キャッシュ対象となる TCP コネクションの検出および横取りを行う。検出したコネクションに含まれる要求がキャッシュ対象の URL であれば、ルータあるいはブリッジ装置はコネクションを横取りしキャッシュサーバに向け直す。キャッシュサーバは自身のキャッシュに指定されたオブジェクトが存在すれば、あたかも本来の WWW サーバから応答が返ったかのように振舞いクライアントにそのオブジェクトを転送する。キャッシュに指定されたオブジェクトがない場合は、クライアントと WWW サーバ間のコネクションはそのままとし転送されるオブジェクトのデータをコピーしキャッシュサーバに格納する。

透過型キャッシュはプロキシ型キャッシュと異なり、クライアントへキャッシュの構成情報を設定する必要がなく、途中の経路上のネットワークに機器を設置しその設定を行うだけでよい。ただし、透過型キャッシュを適切に配置し効果的に動作させるには、ネットワーク管理者はネットワークのルーティングおよび構成について注意深く設計を行っておく必要がある。

2.2.3　先読みキャッシュ

HTML で書かれた WWW ページには多くの埋込み画像およびリンク先への情報が含まれる。その情報を利用して。単にクライアントから要求のあった WWW オブジェクトをキャッシュするだけでなく、要求のあったオブジェクトを元に次にアクセスされるであろうオブジェクトを予測し、キャッシュサーバにおいて能動的に先読みを行うシステムがいくつか提案されている [42]。先読みサーバの 1

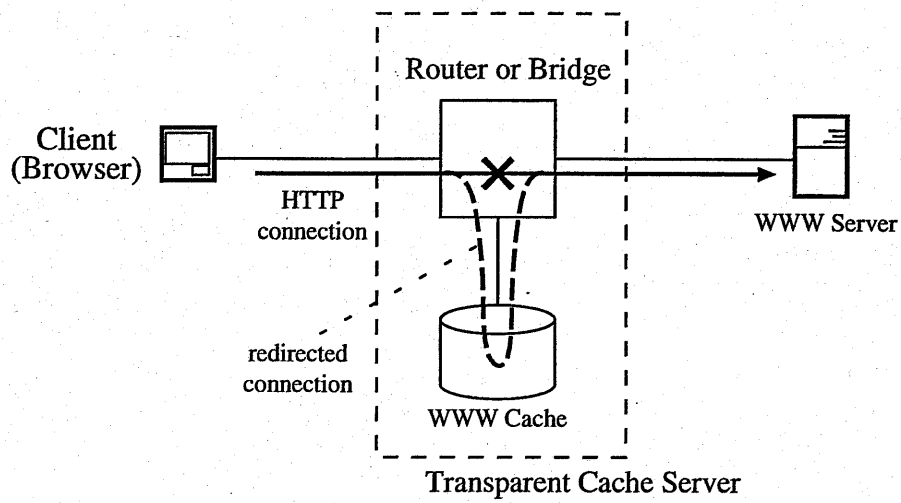


図 2.3 透過型キャッシュ

つである Wcol[43] では、キャッシュサーバ単体で 38%のヒット率が得られるところが、先読みを行うことで 64%のヒット率に改善することができた。ただし、先読みを行わない時に比べてネットワークのトラヒック量は約 3 倍になっており、インターネットとの回線に余裕のない組織ではかえって応答時間の悪化を招くことがあるため注意が必要である。

2.2.4 レプリケーション

レプリケーションとは、ミラーリングとも呼ばれ、データベースの複製を異なるサーバに作成し、複製元のサーバ（マスタ）と複製先のサーバ（レプリカ）の間のデータのコピーをクライアントからのアクセスとは別に非同期で行うものである。WWW サーバのレプリケーションでは、マスタとなるサーバのコンテンツをレプリカのサーバに配布することで、両者のサーバをアクセスした場合に同一のコンテンツを得ることができる。すなわち、アクセス頻度の高い WWW サーバをネットワーク上に分散して配置し、その間でレプリケーションを行うことで、ネットワーク上のトラヒックやサーバ負荷の分散また耐故障性の向上を行うことができる。[44]

アーカイブサイトやフリーソフトの配布用の WWW サーバとして、レプリケーションを適用することが多い。これによって、サーバの負荷分散とトラヒックの地域分散をある程度実現することが可能であるが、コンテンツの配布はサーバの管理者が明示的に指示あるいは事前に設定する必要がある。また、クライアントによる WWW サーバの自律的な選択などもできないため、今後もこのような適用範囲にとどまると考えられる。なお、WWW キャッシュシステムの場合、キャッシュ内容の複製を行うことは、レプリケーションとは呼ばず以下のキャッシュ間のコンテンツの配送と呼ばれることが多い。

複数の WWW キャッシュでオリジナルの WWW サーバのオブジェクトを複製し、プロキシサーバがクライアントからアクセス要求のあったオブジェクトに応じてキャッシュを選択する方式が提案されている [45]。これによると、キャッシュの一部をレプリケーション用に確保することで、キャッシュのヒット率が 4%高くなり、応答時間が 1.5%減少したという効果が得られている。このようにレプリケーションは WWW キャッシュにおいてはそれほど有効ではない。しかしながら、音楽や映像のような時間的にあまり変化しないオブジェクトに対しては有効であり、コンテンツ配送と利用者毎の最適なサーバ選択を世界的な規模で行うようなシステムもある [27]。

2.2.5　コンテンツの配送

WWW キャッシュの内容を他のキャッシュサーバに配送することをキャッシュコンテンツの配送と呼ぶ。キャッシュ内容のうちアクセス頻度の高いものを、他のキャッシュに配送することでヒット率の向上が期待できる。なお、配送によるトラヒック負荷を軽減するために、コンテンツのアクセス頻度を分析してできるだけヒットし易いものを選択することや、回線負荷の低い夜中に配送する方式が提案されている [21]。

WWW オブジェクトの配送には通常 TCP を使ったユニキャストが使用されるが、UDP のマルチキャストを使った配送も考えられる。この場合、信頼性のある伝送プロトコルが必要となり、リアルタイム性は必要ないため RMTP(Reliable Multicast Transport Protocol), MDP(Multicast Dissemination Protocol), MFTP(Multicast File Transfer Protocol) などのプロトコルが適用可能である。また、衛星ネット

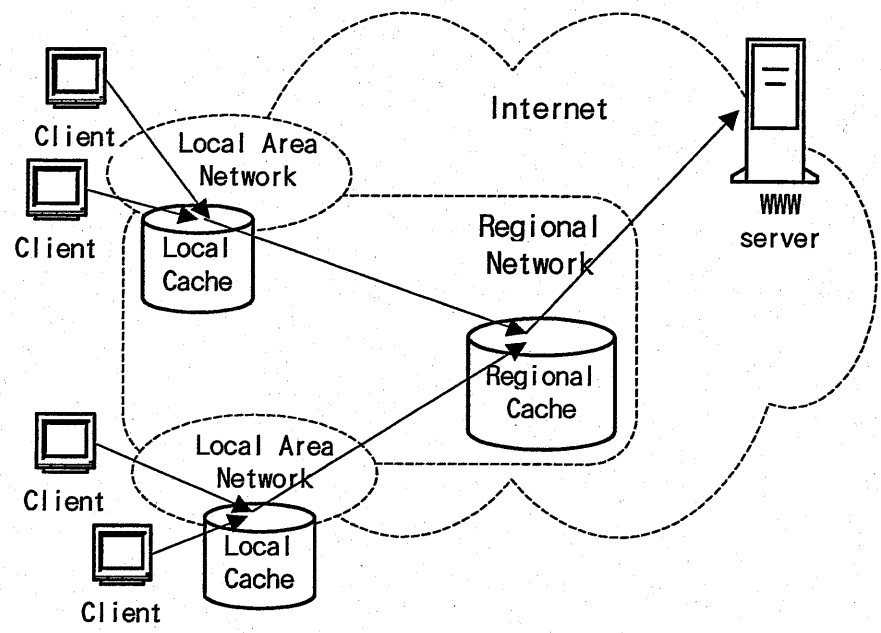


図 2.4 キャッシュサーバの階層構成

ワークにおいては伝送路自体がマルチキャストに対応しており、衛星までの1本のアップリンクを衛星で複数のダウンリンクにすることもできるため、マルチキャスト配送との親和性が高い。

2.3. ICP を使った分散キャッシュシステム

分散 WWW キャッシュシステムの大きな特徴は、インターネット上で階層的なキャッシュ構造を実現できる点にある。図 2.4 に示すように、組織内だけでなくある地域や国などを単位にして階層構造を作ることで、キャッシュサーバの負荷の分散、トラヒックの軽減などが期待される。実際にインターネットで全世界的なキャッシュシステムの実験がすすめられている [46]。キャッシュサーバを協調動作させるには、サーバ間でのキャッシュ情報問い合わせプロトコルが必要となる。このプロトコルとして Squid キャッシュでは ICP（Internet Cache Protocol; RFC2186）[16][47] 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二種類のメッセージから構成されている。

2.3.1　Squid キャッシュ

Squid は米 National Laboratory for Applied Network Research（NLANR）で開発されているフリーな WWW キャッシュサーバである。なお、Squid は、コロラド大の Harvest Project [11] の一環として作られた Harvest Cached がベースになっている。Harvest Cached はその後 NetCache Cached として商品化され、現在は Network Appliance 社の製品となっている [48]。

Squid キャッシュの特徴を以下に挙げる。

- CERN HTTP サーバなどの従来のキャッシュサーバと比べて高速である。
- リクエストごとにプロセスを複製せず、イベント駆動の非同期 I/O を使ってクライアントからの要求を処理するため高負荷に耐えられる。
- 物理メモリとディスクによる 2 階層のキャッシュにより、アクセス頻度の高いオブジェクトはより高速な応答を可能としている。
- WWW サーバの DNS 参照を非同期にし、また IP アドレスをキャッシュすることで、クライアントからみた応答時間を減少している。
- キャッシュ間連係プロトコル ICP を使い、複数のキャッシュサーバで協調して動作することができる。
- 早い段階（1996 年頃）から、SSL（Secure Socket Layer）や HTTP/1.1 に含まれる KeepAlive などの最新のプロトコルに対応していた。

しかしながら、Squid は Unix の単一プロセスとして実装されており多数の要求を処理するためには、TCP 接続受入れの accpet キューの限界、select 関数によるポーリングによる限界、利用可能なファイルディスクリプタの限界などを解決していく必要があるという問題もある。また、Squid キャッシュ自身が、メモリ使用量、ディスクやネットワークの I/O、OS のコンテキストスイッチなどの要因で、ボトルネックになることもあり注意が必要である [49]。

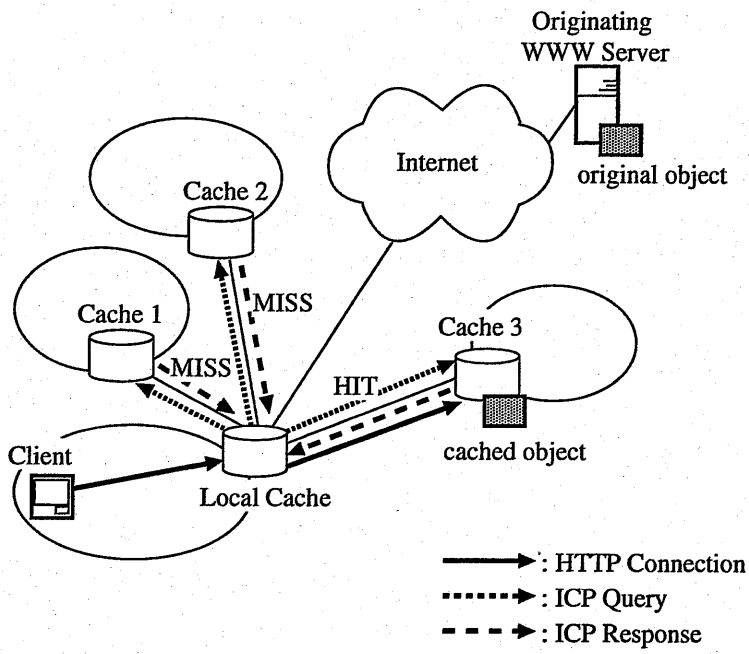


図 2.5 ICP の基本動作

2.3.2 WWW キャッシュ連係プロトコル ICP

ICP（Internet Cache Protocol）とは WWW キャッシュサーバ間でのキャッシュデータのやりとりを行うためのプロトコルである。現在は RFC2186 で定義された ICP Version 2 が広く利用されている。

図 2.5に ICP の基本動作の概念図を示す。ICP の基本動作は以下のような手順になる。

1. キャッシュサーバから隣接キャッシュサーバへ、クライアントから要求された URL を指定して ICP Query メッセージを出す。

2. ICP Query メッセージを受信したキャッシュサーバは、その URL で指定されたオブジェクトが自身のキャッシュの中にあれば HIT、なければ MISS の情報を ICP Reply メッセージとして返す。

3. キャッシュサーバは HIT が返って来たキャッシュサーバから、HTTP でオブ

ジェクトを取得し、クライアントに転送する。あるいは、全てのキャッシュサーバから MISS が返るか、タイムアウトと判定すると、本来の 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。

Squid キャッシュでは、設定ファイルで指定した複数の隣接キャッシュサーバに対してこの ICP による問い合わせを行い、HIT の応答メッセージが返ってきたキャッシュサーバから HTTP でオブジェクトを転送している。また、キャッシュサーバ選択のアルゴリズムとして、以下のような処理も行い、ヒット率の増加、応答時間の短縮などを図っている。

- 複数の隣接キャッシュサーバから HIT の応答メッセージが返ってきた場合は、もっとも早く HIT を返してきたキャッシュサーバを利用する。
- 隣接キャッシュサーバに明示的に重み付けを行って、静的に優先度を付与することができる。
- いずれの隣接キャッシュサーバからも HIT 応答メッセージが到着しなかった場合は、事前の静的設定によって本来の WWW サーバまたは指定された親キャッシュサーバを使用する。

図 2.6に、ICP を使った Squid キャッシュサーバを階層的に配置したシステム例を示す。Cache1,2,3 は相互に隣接キャッシュの関係（以下、sibling）になり、Cache4 はそれらの親キャッシュの関係（以下、parent）になる。

2.4. WWW クライアントにおけるプロキシの構成管理

初期の WWW クライアントでは、アクセスしようとするオブジェクトの HTTP, FTP, Gopher, WAIS などのプロトコル別に、プロキシサーバの名前または IP アドレスと TCP ポート番号を指定する必要があった。1996 年に公開された Netscape Navigator ver.2 において、PAC（Proxy Auto-Config）というプロキシサーバの選択を自動化する仕組みが導入された。さらに、PAC 記述を自動探索する方式

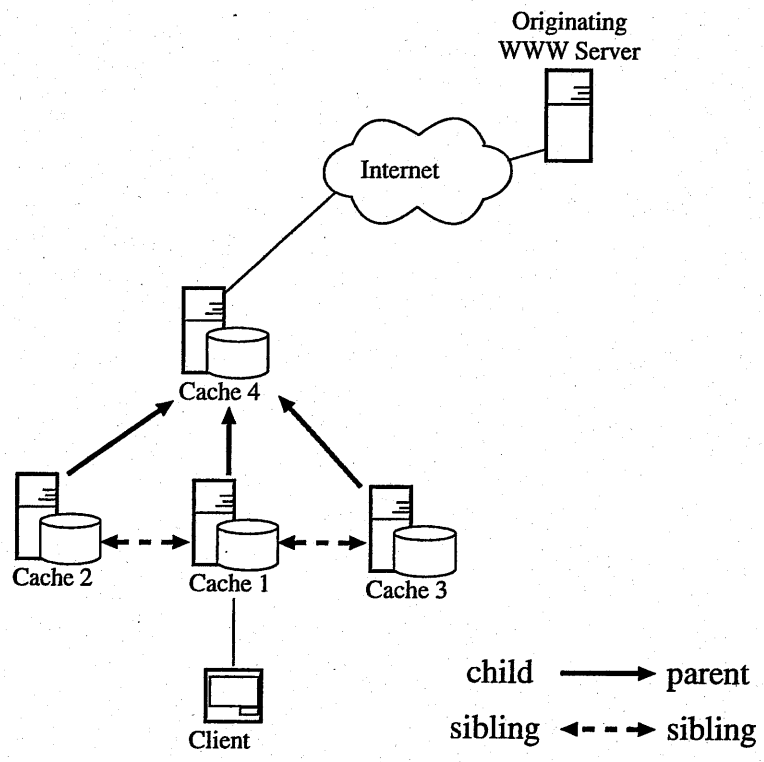


図 2.6 ICP を使った階層型分散キャッシュシステム

として、1999 年に Internet Explorer ver.5 において、WPAD（Web Proxy Auto-Discovery Protocol）と呼ばれる WWW プロキシ用の資源探索プロトコルが実装され、クライアンおける構成管理を自動化することが可能となった。

なお、透過型キャッシュサーバを使ったシステムでは、WWW クライアントから WWW サーバへの要求は透過的であり、キャッシュの存在をクライアントが把握するはなく、クライアントにおいて特別な設定は必要ないため本節では言及しない。

2.4.1 プロキシの手動設定

代表的な WWW クライアントであった NCSA Mosaic では、HTTP や FTP などのプロトコル別にプロキシサーバのアドレスと TCP ポート番号の設定を手

動で行う必要があった。Unix 版では環境変数を参照することも可能であったが、基本的にユーザがプロキシサーバの情報を入手しクライアントに設定する必要があった。現在広く使用されているクライアントは全てプロキシサーバの手動設定を行うインタフェースを備えている。

この方式では、利用者はプロトコル別のプロキシサーバの設定情報を事前にネットワーク管理者から入手しておく必要があり、その情報を正確にクライアントに設定しなければならない。よって、設定間違いによる障害が発生する可能性は避けられず、またプロキシサーバの指定は 1 つしか記述できないため耐故障性が低い、ネットワークの変更に伴なって最適な構成を保つことが困難であるという問題がある。

2.4.2　プロキシの自動設定記述

前述のように WWW クライアント（ブラウザ）のプロキシサーバ設定を手動で行う場合、プロトコル毎のプロキシサーバ名とポート番号を利用者が事前に入手し設定しなければならない。このプロトコル毎のプロキシサーバ名とポート番号の設定を自動化するための仕組みが、Netscape 社によって提案された PAC である [50]。PAC では、アクセスしようとする URL に対してプロキシサーバを選択する手順を、JavaScript を使った簡単な言語で記述することができる。また、PAC の記述ファイルはあらかじめネットワークの管理者によって作成されたものを、各クライアントに対して WWW サーバを使って提供を行う。この PAC ファイルの URL は、利用者がクライアント毎に手動で設定する必要があり、完全に自動化されているわけではない。この点については次の WPAD の項で述べる。

PAC は、自身の IP アドレスおよびアクセスしようとする WWW オブジェクトの URL から、プロキシ参照の動作を決定する評価関数であり、クライアントがオブジェクトをアクセスする際に呼び出され、その結果をもとにクライアントはオブジェクトのアクセスを実行する。プロキシの手動設定の場合と比較した PAC の利点は、(1) 設定項目が PAC の所在を表す URL だけとなる、(2) プロキシサーバの選択をネットワーク管理者側で制御できる、(3) クライアントの IP アドレスとアクセスしようとするオブジェクトの URL によってプロキシサーバを選択することが可能となる、(4) 評価関数の結果に複数のプロキシサーバを指定できる

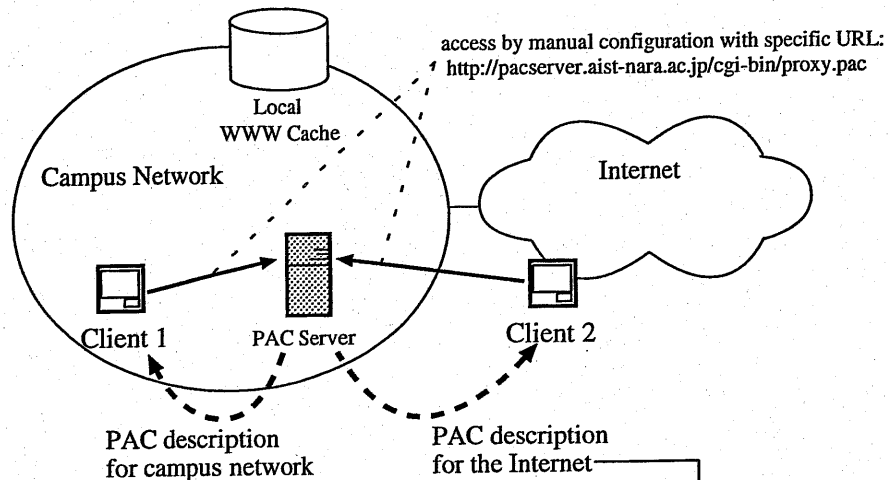
ため耐故障性に優れる、などが挙げられる。

PAC の記述例として、図 2.7に奈良先端科学技術大学院大学で構築したプロキシ自動設定環境を示す。ここでは、ファイアウォールの内部である学内に位置するクライアントと、学外いわゆるインターネットに位置するクライアントに対して配布する PAC を切り替えるようになっている。学外のクライアントに対しては学内のプロキシサーバを利用せず、クライアントが直接オブジェクトにアクセスするように PAC を記述している。学内のクライアントに対しては、アクセス先が学内であるかプロトコルが HTTPS（Secure HTTP）であれば直接アクセスを行うようにし、そうでなければプロキシサーバを経由するようにしている。また、HTTP アクセスについては当時実験を行っていた先読みプロキシサーバを経由するように制御している [41]。このように PAC を使用することで、ネットワーク管理者によって URL とプロトコル別に柔軟なプロキシサーバの設定を行うことができる。

また、PAC は JavaScript 言語によって比較的自由に記述できるため、アクセスしようとする URL によってプロキシサーバを変更したり、簡単な負荷分散を行うことも可能となる。この点については 2.6.2章で述べる。

2.4.3 プロキシの資源探索手法

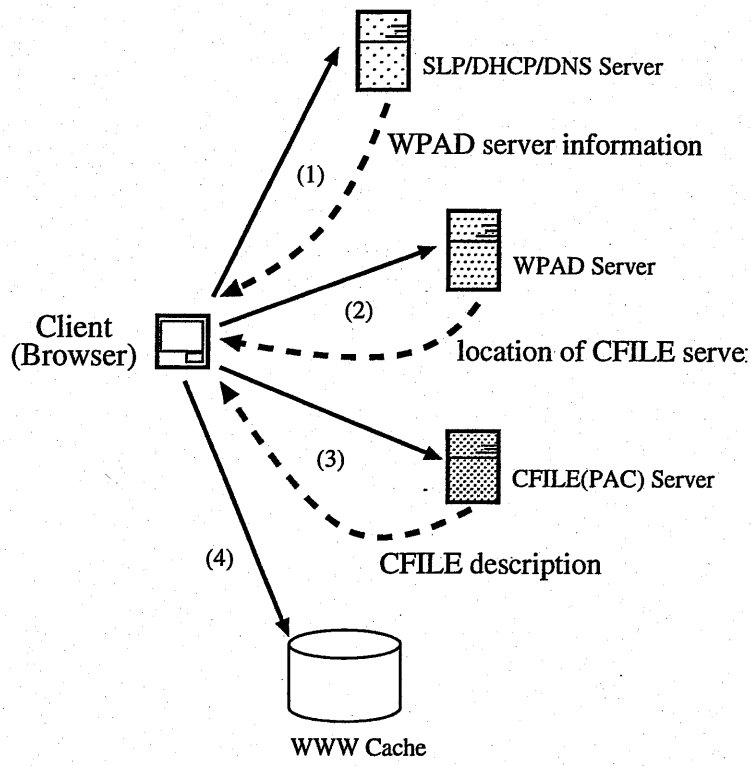
PAC サービスを利用することで WWW の利用者は、キャッシュサーバや現在のネットワークの位置情報を意識することなく、クライアントのプロキシ設定をキャッシュサーバあるいは所属するネットワークの管理者の意図どおりに設定できるようになる。しかしながら、PAC サーバあるいはそのサービスの資源探索（Resource Discovery）手段が提供されていないために、利用しようとしているネットワーク（場所）に応じた PAC サービス設定を、利用者がクライアントの設定として手動で入力せざるを得ないという大きな問題がある。これは SLP（Service Location Protocol）[51][52] などの資源探索プロトコルによる解決が望ましい。しかしながら、現在のところ SLP による実装はなく、WPAD という WWW プロキシに特化した形の資源探索方式の提案が行われており、現在 Internet-Draft の段階である [19]。なお、現在 WPAD を使ったプロキシの自動設定が搭載されているクライアントとしては Microsoft 社の Internet Explorer Ver.5 がある。




```
// Proxy Automatic Configuration of Netscape Navigator / MS-IE
//    for NAIST Mandara Network (inside firewall)
//
//  1997.09.16 v1.3 h-inoue@is.aist-nara.ac.jp

function FindProxyForURL(url, host) {
 var myip = myIpAddress();
 var proxy_squid = "PROXY wwwproxy.aist-nara.ac.jp:8000;" +
     "DIRECT";
 var proxy_prefetch = "PROXY wwwproxy.aist-nara.ac.jp:9000;" +
     "PROXY wwwproxy.aist-nara.ac.jp:8000;" +
     "DIRECT";

 // if in NAIST network, then it will access directly
 if (isPlainHostName(host)
  || shExpMatch(host, "*.aist-nara.ac.jp")
  || shExpMatch(host, "163.221.*")) {
  return "DIRECT";
 }

 // skip WAIS request
 if (url.substring(0, 5) == "wais:") {
  return "DIRECT";
 }
 // skip https
 if (url.substring(0, 6) == "https:") {
  return "DIRECT";
 }
 // if no anonymous FTP, then it will access directly
 if (url.substring(0, 4) == "ftp:" && shExpMatch(host, "*@*")) {
  return "DIRECT";
 }

// only HTTP to prefetching server
 if (url.substring(0, 5) == "http:") {
  return proxy_wcol;
 }

 // if everything else, then it will use normal proxy server
 return proxy_squid;
}
```

```
// PAC for the host outside NAIST Mandara network

function FindProxyForURL(url, host) {
 return "DIRECT";
}
```

図 2.7　PAC サービスによるプロキシ自動設定の例



(1) discovery for WPAD service with SLP, DHCP, or DNS
(2) query for CFILE location to WPAD server
(3) retrieval of CFILE description from CFILE server
(4) cache access according to the CFILE description

図 2.8 WPAD によるプロキシ設定の探索手順

WPAD では、サービス資源の探索手順を定義しており、クライアントは図 2.8 に示す手順でプロキシサーバの情報を獲得する。WPAD におけるプロキシ設定の記述は CFILE（Configuration File）と呼ばれ記述の定義は PAC と全く同様である。また、それを提供する URL のことを CURL（Configuration URL）と呼ぶ。WWW クライアントは DHCP, SLP, DNS などを使って CURL を入手する。あるいは、簡単な実装として “http://wpad + クライアントの所属するドメイン名/wpad.dat” という URL を CURL とみなして上位のドメインにさかのぼって検索していく方式が提案されている。CURL から CFILE が取得できた場合、クライアントはその記述に従ってプロキシ設定を行う。

2.5.　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なければならない。プロキシ型のキャッシュサーバでは、各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 ポート番号のような構成情報を適切に設定し、またネットワーク環境の変化に追従して更新していく必要がある。透過型キャッシュサーバでは、クライアントからのコネクションを横取りするためのルータあるいはブリッジ装置において、キャッシュサーバのホスト名または IP アドレスを設定する必要がある。

2.5.1　プロキシ型キャッシュサーバの構成管理

協調して動作する分散キャッシュサーバの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 [53] 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全てに届く。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 [20]。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[54] 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。

ICP のような協調動作のためのメッセージを使用せずに、キャッシュ全体のヒット率を高く保つことを意図した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ための動作が記述できること [50] を利用して、アクセスしようとするオブジェクトの URL 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 [29][55]。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ヒット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のはブラウザに限られており、複数のキャッシュサーバから成る階層的なキャッシュシステムを自動的に構成することはできない。

Squid の開発チームによって、Cache Digests という、ICP を使用せず、キャッシュの内容を圧縮したものをキャッシュサーバ同士で HTTP でやりとりし、ICP による遅延と消費帯域を減らす方式が提案されている [56]。しかしながら、他のキャッシュサーバの内容を記憶するために、キャッシュサーバのメモリがより必要となるという問題点がある。

また、各キャッシュサーバにおける過去のアクセス記録をもとにして、キャッシュサーバでどのオブジェクトをキャッシュするかを事前に決定し、その情報を共有することでキャッシュ全体のヒット率を向上させる方式も提案されている [57]。

2.5.2 透過型キャッシュにおける構成管理

WCCP (Web Cache Communication Protocol) [58] は、シスコのルータと同社の透過型キャッシュサーバ CacheEngine[59] が連携するためのプロトコルである。透過型キャッシュにおいてはルータあるいはブリッジがキャッシュすべき HTTP などの TCP セッションを横取りし、キャッシュサーバへそのセッションを転送する際に、WCCP を使用してキャッシュサーバをクラスタ化しアクセスを転送するキャッシュサーバを決定している。また、耐故障性を備え、キャッシュの変更などにも対応できるようになっている。

2.6. WWW キャッシュの負荷分散

WWW キャッシュサーバへのアクセスの集中によるサーバ自身の反応時間の増大が問題になっている。キャッシュサーバは 1 秒あたり 100 から 1000 ものアクセスの処理が要求される。そのため、複数のキャッシュサーバをクラスタ化することや、特殊なネームサーバを使って負荷が最低であるキャッシュサーバの IP アドレスを返すことでサーバの負荷分散を行う研究などが行われている [60]。また、接続相手の種類によってサーバが提供するサービスを切り替えるような、サービススイッチ機構に NAT を利用した研究もある [61] 。

2.6.1 DNS を使ったラウンドロビン方式

DNS（Domain Name System）はインターネット上で名前を解決するための分散データベースで、ホスト名から IP アドレスやメールサーバを検索したりするのに使用されている。DNS では 1 つのホスト名に対して複数の IP アドレスを指定することができ、その場合、DNS は問い合わせがある度にその複数の IP アドレスをラウンドロビンで入れ換えて返す仕組みになっている。この仕組みを使用して、あるプロキシサーバのホスト名に対して実際は複数のプロキシサーバの IP アドレスを DNS に設定しておくことで、クライアント毎にラウンドロビンでプロキシサーバを選択させることができる。なお、DNS の SRV レコードを使った優先度付きのラウンドロビンでサービスを探索する方式については RFC2052[28] で提案されている。

DNS を使用する方式では、事実上クライアント毎にプロキシサーバが決定されてしまうため、クライアント間のアクセス相関によるキャッシュヒット率の向上を実現するには、前述の ICP などを使って複数のプロキシサーバ間でキャッシュの協調動作をさせることが不可欠となる。すなわち、全てのキャッシュ間で問い合わせおよびその応答のメッセージを送受信する必要があり、次に説明するクライアント側で URL に応じて決定的にプロキシサーバを選択するハッシュルーティング方式のほうが大規模システムには適している。ただし、DNS を使用することで、ネットワークあるいはプロキシサーバの管理者で負荷分散設定を制御でき、またクライアントおよびその利用者は設定を特に意識する必要はない。

なお、DNS を使ったラウンドロビン自体の問題点として、(1) クライアントの名前解決ライブラリ側での制限または DNS パケット長の制限により、数十程度のエントリしか持つことができないため大規模な運用には向かない、(2) サーバやネットワーク状態などに応じた分配ができないため、あるサーバが過負荷あるいは停止している時でも DNS は問い合わせに対してそのサーバのアドレスを返してしまう、(3)DNS の設定の変更を行っても、その変更がインターネット上に伝播するのには数時間から数日かかるため、構成の急な変更に対応できない、などが挙げられる。同様の DNS を使った負荷分散方式は WWW サーバでも適用可能であるが [62][63]、上記のような問題は共通である。

2.6.2 Hash Routing

URL あるいはその一部をキーとしたハッシュ関数の出力をもとに、クライアントが決定的にプロキシ（キャッシュ）サーバを選択する方式である。同じハッシュ関数を使用することで、異なるクライアントからであっても、同じ URL を持つ WWW オブジェクトに対しては同じキャッシュサーバにアクセスに行くため負荷分散を行いながら高いキャッシュヒット率が期待できる。クライアントは前述の PAC を使用して JavaScript でハッシュ関数を記述し、アクセスすべきキャッシュサーバを決定する [55]。また、Microsoft 社による CARP（Cache Array Routing Protocol）[30] では、ハッシュ関数を用いて URL をグループ分けし、キャッシュサーバ間にアクセスを振り分ける仕組みが提案されている。

この手法を、キャッシュサーバが他の隣接あるいは親キャッシュサーバを選択する際に適用し、ICP を使ったキャッシュサーバ選択の代替として使用し、ICP トラヒックの減少と応答時間の短縮が実現できることも報告されている [64]。

2.6.3 NAT あるいは 4 層スイッチによる負荷分散

WWW サーバあるいはキャッシュサーバへのアクセスを分散させるための機構として、NAT(Network Address Translation/Translator)[65] を使った実装がある。複数のホストの集合から成るサーバを、NAT によって 1 つの IP アドレスを持つホストに見せる。WWW クライアントからのアクセスはその単一の IP アド

レスによって行われ、NATによって現在の負荷が最小であるサーバあるいは単純にラウンドロビンで順番に振り分けることで負荷分散を実現できる。現在は、4層スイッチという名前で商用の製品が入手できるが、その原理は同様である。ネットワークモデルにおける4層であるUDPおよびTCPのポート番号毎にセッションをスイッチさせることができることから、このように呼ばれている。

筆者は1996年にサービスを分散させる機構としてLB-NAT(Load Balancing Network Address Translator)と呼ぶ負荷分散機構の実装を提案した。NAT技術を用いることでサーバプール機構を実現し、複数のホストすなわち複数のIPアドレスからなるWWWサーバ群を、1つのIPアドレスを持つホストに見せかける。これにより、WWWクライアントからのアクセスは、その単一のIPアドレスによって行われ、現在の負荷が最小であるWWWサーバに振り分けることで負荷分散を実現する。本実装ではWWWクライアントからのアクセスは完全に透過的である。また、見かけ上のWWWサーバの耐故障性を向上することも可能となる。同様の手法がLSNAT（Load Sharing using IP Network Address Translation）として1998年にRFCとして提案されている[66]。

2.7. むすび

プロキシサーバをもとに開発されたWWWキャッシュシステムは、複数のキャッシュサーバが協調動作する分散キャッシュシステムとして現在も広く使用されている。WWWクライアントであるブラウザ、また分散キャッシュシステムにおいて、キャッシュの構成定義をいかにしてして設定し、またその正しい設定状態を維持するにはどうすればよいのかという問題がある。透過型のキャッシュシステムはでは前者の問題は原理的に発生せず、後者については管理者が設定した内容をもとにWCCPを使用してキャッシュサーバの選択を行うことができるが、現時点ではまだ未熟な方式であると考えられる。プロキシ型のキャッシュシステムにおいては、前者の問題はWPADという資源探索プロトコルを使用して自動的に設定が可能である。ただし、WPADをサポートするブラウザが限られているため今後の普及が望まれる。後者の問題を解決する手法がいくつか提案されているが、あるキャッシュサーバの他のキャッシュのキャッシュ内容を把握できないこ

とが大きな問題であり、これを解決することで効率的な分散キャッシュの自律分散化が実現できる。

また、分散キャッシュシステムにおいて他のキャッシュサーバを選択する方式にも改善の余地がある。現在は、ICPが最も早く返ってきたサーバを選択する方式や、アクセス情報の一部をハッシュ関数によって計算し決定的にサーバを選択する方式などが適用されている。それに加えて、キャッシュサーバからWWWサーバや他のキャッシュサーバに至る経路やネットワークの状態を考慮してオブジェクトの取得先の選択を行う方式や、利用者の要求を統計的に分析してオブジェクトを先読みしてキャッシュのヒット率と応答時間を改善する方式などが考えられる。これらの方式を適用することで、キャッシュのヒット率を向上させ、また取得時間を減少することが可能である。

第 3 章　キャッシュサーバの自律分散化

分散キャッシュシステムの自律的動作を実現するために、ヒントサーバを導入した分散WWWキャッシュシステムの構成自動化手法について提案し、そのアーキテクチャと実装について述べる。また、従来のシステムを理想的な状態にした場合とキャッシュ全体のヒット率やトラヒックおよび応答時間を比較し、提案システムの効果を評価する。

3.1.　まえがき

分散キャッシュシステムでは、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、管理者に常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。

本章では、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 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、従来システムと比較をすることで行う。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場合の、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。

3.2. WWW キャッシュサーバの構成情報の管理

3.2.1 分散キャッシュサーバ

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を図 3.1に示す。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全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。

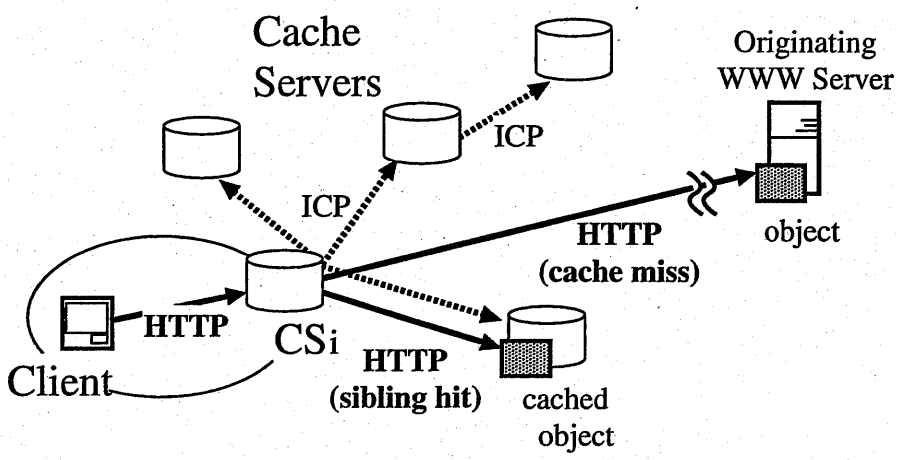


図 3.1 ICP を使った分散キャッシュシステムの基本構成

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。

3.2.2 分散キャッシュサーバの構成管理

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。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。

例えば、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る毎に、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。

3.2.3　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上記のような問題は、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[67]。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、既にコンピュータの資源を多く必要としていることから[41]、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。

本章では、他のキャッシュの情報を管理をキャッシュサーバから分離し、システムの自律分散化を実現する方式を提案する。実際には、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常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[68] [69]。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

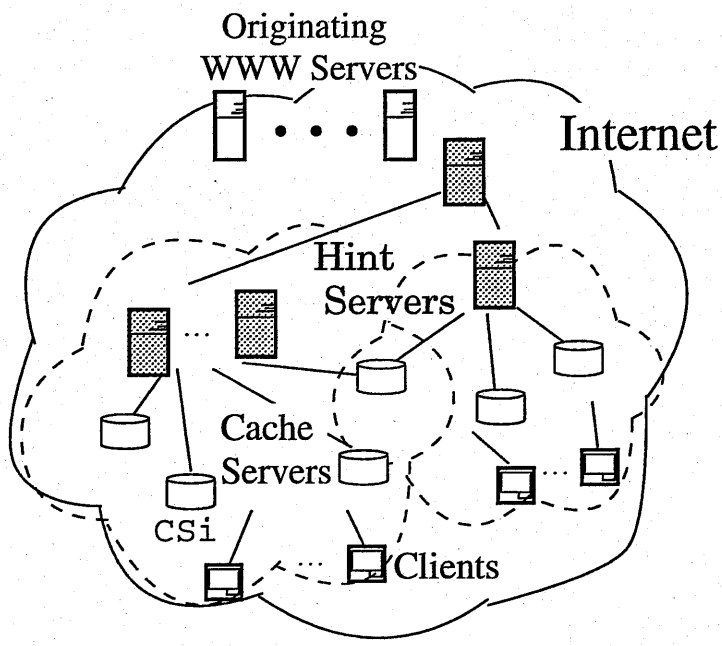


図 3.2 提案するシステムのモデル

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。

3.3. ヒントサーバを使った分散キャッシュシステム

3.3.1 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 3.2 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。

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、例えば 1 台のヒントサーバからの応答が無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全く無くなった場合でも、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。[41]

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 ms から数十 ms 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。

3.3.2 システムの動作とプロトコル

3.3.2.1 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通りである。

- 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。

- ヒント問合わせメッセージ（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無い場合にヒントサーバに問合わ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キャッシュサーバ自身の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。

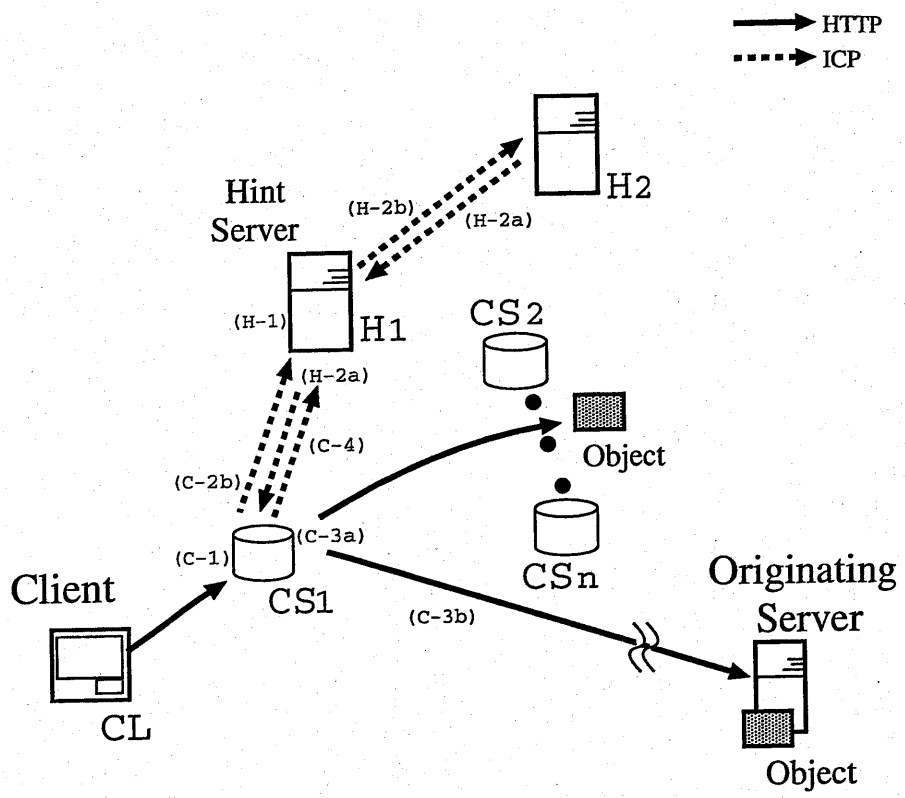


図 3.3 提案システムにおける WWW アクセス手順

3.3.2.2 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。(図 3.3)

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。

(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。

(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、以降の問合わせを行わない。利用できなくなった H は一定時間ごとに HQUERY を送信し、HREPLY を受信した時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 3.3）

(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。

3.3.2.3　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、自身の持つ全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全て削除する。

3.3.3　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わ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わせの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わせの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20byte[16]から80byte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ーヘッドが加わる。

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

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

たとえば、$n=5, h=1, a=0.4, b=0.1, L=50$（図3.5(a)）の場合、$I_c = 5N, I_h = 2.2N, T_c = 350N, T_h = 286N$となる。この場合、提案システムでは従来システムと比較して、ICPメッセージ数は約4割に、トラヒックは約8割になる。よって、提案システムでは、ICPに起因するネットワーク負荷は低くなることが期待できる。

3.4. 実装と評価

3.4.1 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3.4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。

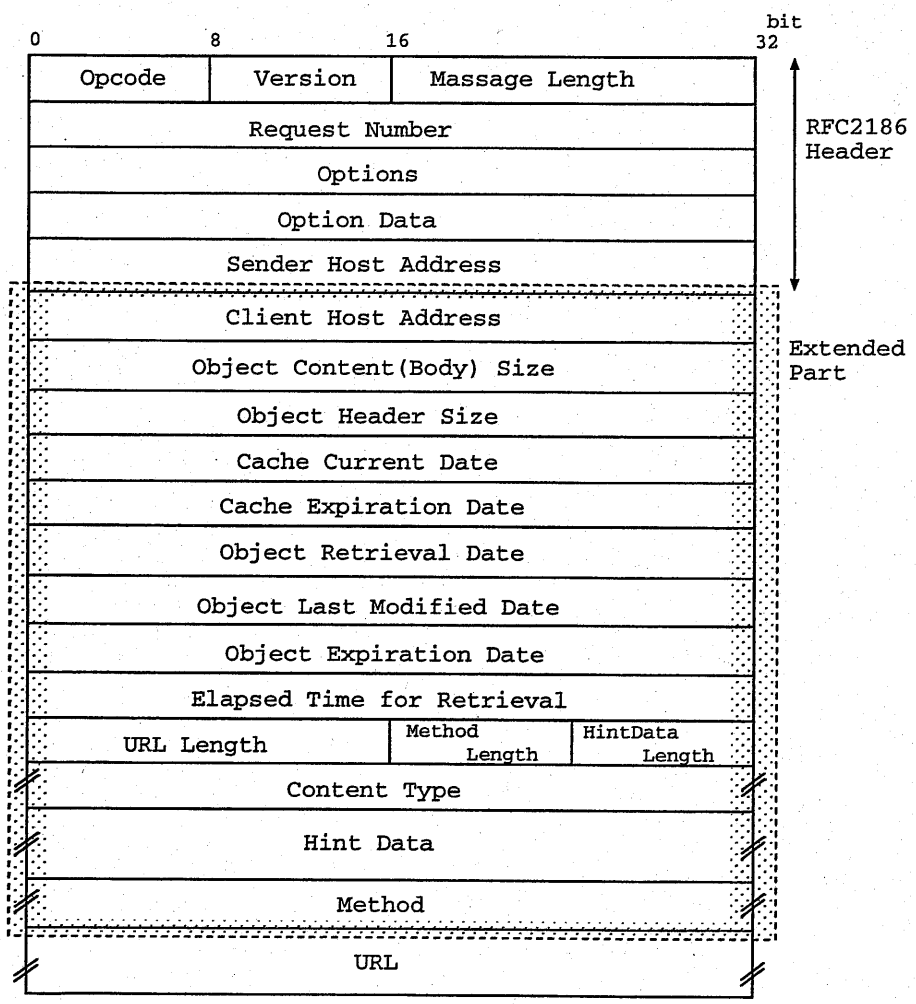


図 3.4 追加したICPのフォーマット

表 3.1 キャッシュサーバの構成

name	value
Computer model	SGI O2
Operating system	IRIX6.3
Phyical memory	96 Mbytes
Network I/F	100Base-TX
Cache parameter	**value**
WWW cache memory	8 Mbytes
WWW cache disk	70 Mbytes
ICP timeout	1 second

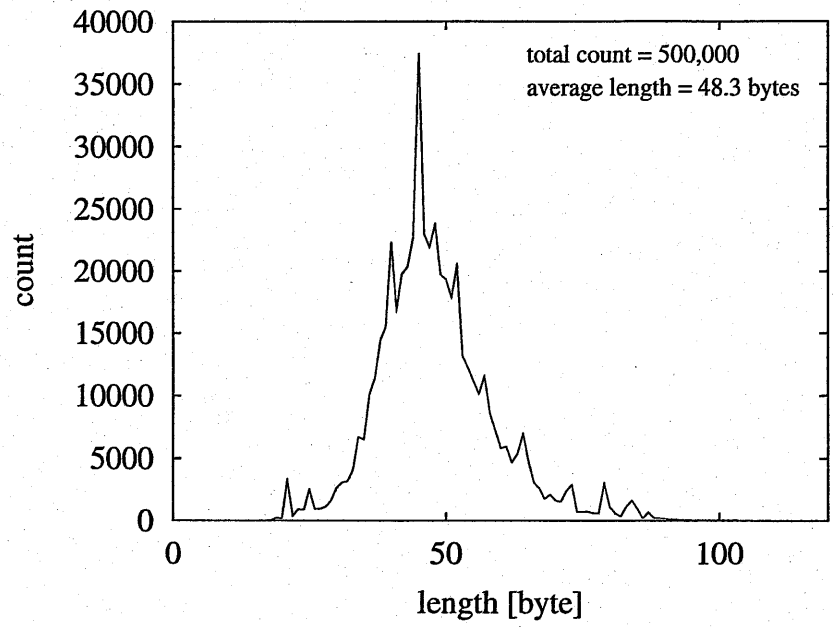
3.4.2 評価

3.4.2.1 実験内容

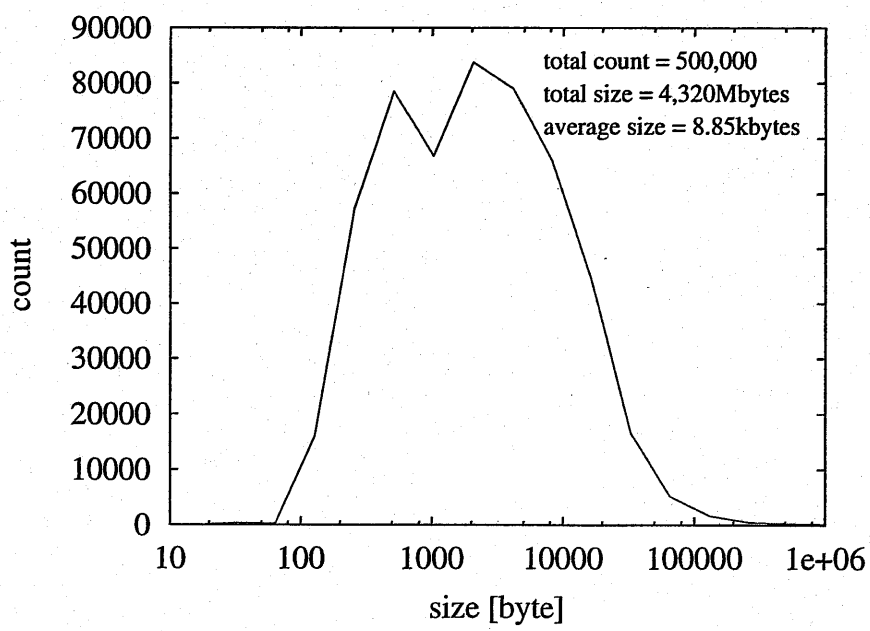
今回提案した方式に基づくキャッシュシステムの評価として、従来の分散キャッシュシステムとの比較を行った。実験は、図 3.6のように両システムの典型的な構成を作成し、クライアントには実際のキャッシュサーバで運用中に記録したアクセスログを入力し、実際の WWW サーバにアクセスしてシステムの性能を計測した。アクセスログには奈良先端科学技術大学院大学で平日 5 日間の間に学内の利用者からアクセスされた内容のうち、キャッシュ可能な HTTP のメソッドである GET リクエストを抽出したものを使用した。アクセスログは 50 万件分で、その URL の長さの分布や平均長、およびオブジェクトのデータサイズとその分布は図 3.5に示すとおりである。使用したキャッシュサーバの構成を表 3.1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) キャッシュ構成定義の内容、(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

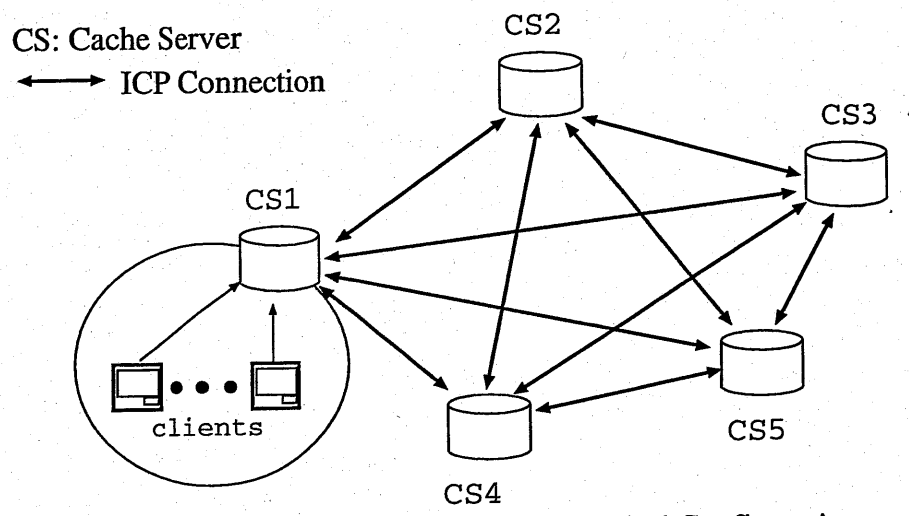


(a) URL Length Distribution

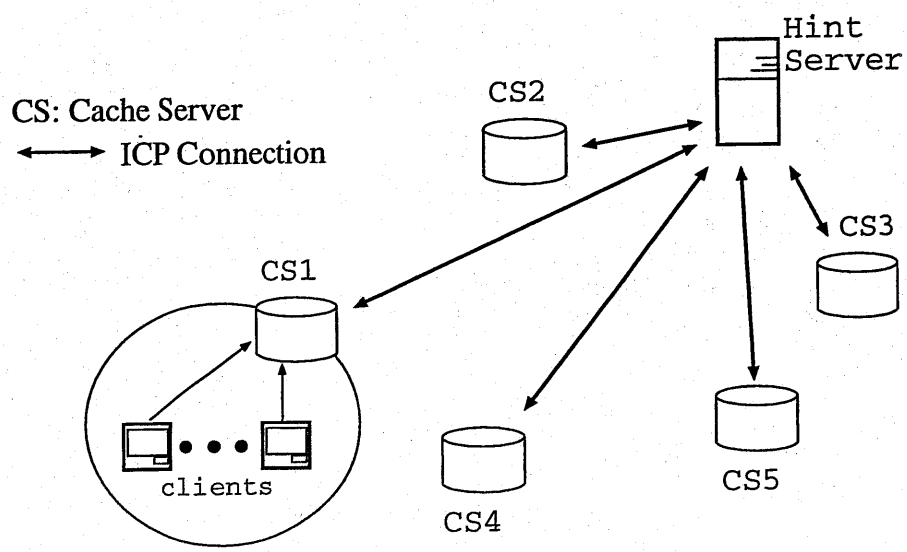


(b) HTTP Data Size Distribution

図 3.5 実験に用いたアクセスログの統計情報



(a) Conventional Cache System on Ideal Configuration



(b) Proposed Cache System

図 3.6　実験システムの構成

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全く同じではないため、実験毎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。

3.4.2.2 実験結果

（1）キャッシュの構成定義

各キャッシュサーバにおける、キャッシュの構成定義の内容（抜粋）は図3.7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全て設定が異なり、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、全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3.6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

```
#          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 3.7　キャッシュ構成情報の比較

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同士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 3.6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、また全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。

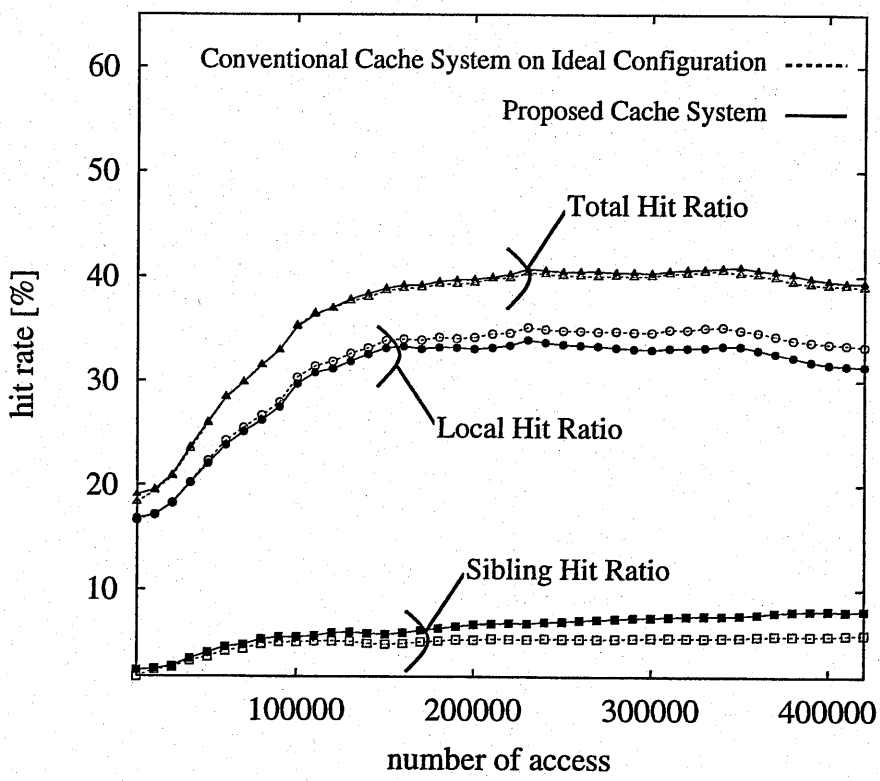


図 3.8 キャッシュヒット率の比較

(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 3.6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 Ratio) と ICP による隣接キャッシュへのヒット率 (Sibling Hit Ratio) およびそれらの和 (Total Hit Ratio) を測定したものは図 3.8のようになった。なお、図 3.6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。

3.4.3　考察

3.4.3.1　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、3.4.2.2節の結果（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。

3.4.3.2　ICPのメッセージ数とトラヒック

表3.2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3.3.3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。バイト数ベース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。特に、HREPLYメッセージに関しては、拡張した部分のうちヒント情報のフィールドのみ使用しており、URLとオブジェクトの情報に関するフィールドを省略可能である。

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 3.2 ICP の数とトラヒックの比較

	Conventional System	Proposed System
Total ICPs	270 $\times 10^4$	129 $\times 10^4$
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)
Total Hit Ratio	37.9 %	38.0 %
HTTP Traffic	1,250 MB	1,230 MB
Sibling Hit Ratio	5.5 %	7.5 %
HTTP Traffic	150 MB	224 MB
Miss Ratio	62.1 %	62.0 %
HTTP Traffic	3,070 MB	3,080 MB

mps: messages per second MB: Mbytes

（表の Sibling Hit の HTTP Traffic）と同程度に相当していることがわかった。

3.4.3.3 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 180MHz; Memory 96MB）と PC Unix（BSD/OS; CPU 486DX2 66MHz; Memory 32MB）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 3.9 に示す。ワークステーションにおける応答時間は、1 から 2ms の間で安定していることがわかる。PC Unix も 3ms 以内にほぼ収まっているが、20 万アクセスを超える頃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

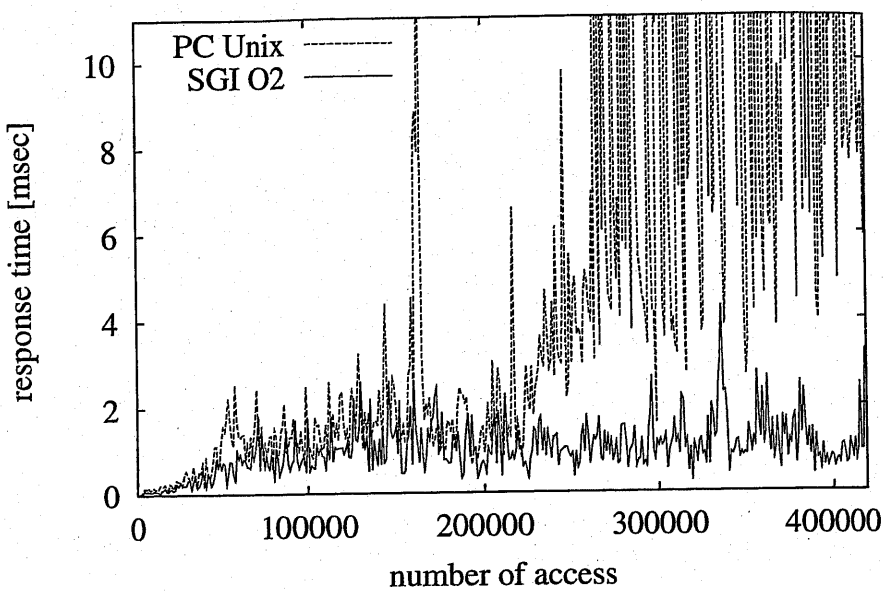


図 3.9 ヒントサーバの ICP 応答時間

記憶のスワップを起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 2ms 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わかった。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。例えば、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。

3.4.3.4　ICP のタイムアウト率

実験を行った図 3.6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 3.10に示す。なお、表 3.2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

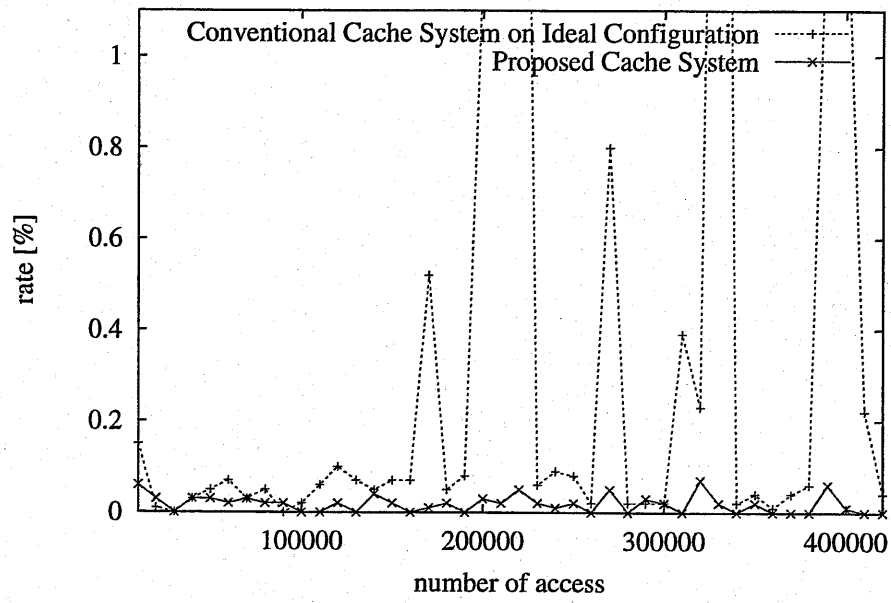


図 3.10 ICP のタイムアウト率

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同士の 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わかる。文献 [41] 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。

3.5. むすび

本章では、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。

第 4 章 衛星ネットワークにおける適応型キャッシュ

静止衛星を使ったネットワークでは、リムに位置する国からのアクセスは衛星回線を経由するため回線自体のRTTである約500msが大きな遅延の原因となる。また、衛星回線は帯域あたりのコストが非常に高価であり、その衛星回線がボトルネックになりがちである。WWWのトラヒックを削減するためには、単純にリム側にWWWキャッシュを設置することである程度の改善は可能であるが、実際には衛星回線のトラヒックの制御と衛星回線のRTTに起因する応答時間の減少をいかに行うかが問題となる。AI3（Asian Internet Infrastructure Initiative）ネットワークは、アジアの複数の研究機関を衛星リンクで結んだ研究ネットワークで奈良先端科学技術大学院大学を中心として運営を行っている。AI3は日本をハブ、他の国をリムとしたスター状のトポロジーで構成されている。インターネットバックボーンとの接続は、日本側で100Mbit/s以上の速度でWIDEのバックボーンに、他の国は双方向1.5Mbit/sの静止衛星リンクで経由で日本のハブ局に接続されている。本章では、衛星ネットワークの特性を考慮したWWWキャッシュ制御方式の設計と、AI3ネットワークへ適用した実験結果について述べる。

4.1. まえがき

AI3（Asian Internet Infrastructure Initiative）は、アジアの複数の研究機関を衛星リンクで結んだ研究実験用のバックボーンネットワークである。AI3ネットワークは日本を中心（以下、ハブ）、他の数カ国を周辺（以下、リム）としたスター状のトポロジで構成されている。リムに位置する国からのアクセスは静止衛

星回線を経由するため回線自体の RTT（Round Trip Time）の約 500ms が大きな遅延の原因となる。また衛星回線は、1.5Mbit/s とハブ側のバックボーンの帯域幅の 100Mbit/s と比較し低速でありボトルネックの原因となりやすい。そこで、衛星回線の下流すなわちリム側に WWW キャッシュを設置することは、衛星回線のトラヒックの削減とリム側の利用者からみた応答時間の短縮に有効となる。

本研究では、ハブ側のキャッシュはバックボーンの帯域に余裕があるため、リム側からの要求に応じると同時にコンテンツの先読みを行い、同時に大容量のキャッシュディスクを備えキャッシュのヒット率を向上させる戦略をとる。リム側のキャッシュは同様に先読みを行うことで、ヒット率の改善を図るが、利用者のアクセスの傾向を考慮せず無制限に先読みを行うとトラヒックの大幅な増加をまねき衛星回線がボトルネックとなってしまう。そこで、サーバまでの RTT や、利用者のアクセスパターンをもとに参照される確率の高いオブジェクトを決定し優先度をつけることで、ヒット率を損なうことなく、先読みによるトラヒックを低減することを可能とする。また、衛星ネットワークのトポロジがスター状であることを利用し、マルチキャストを使ってオブジェクトを複数のキャッシュに同時に配送することで、キャッシュのヒット率の向上と衛星回線の帯域の消費を抑制する。

このような衛星ネットワークの特性を考慮した WWW キャッシュ制御の方式を提案し、その適用および評価を行った。

4.2. 衛星ネットワークにおける WWW キャッシュシステム

4.2.1 AI3 プロジェクト

AI3 では、アジアの研究機関を結ぶインターネットバックボーンを提供すると同時に、それを研究基盤としていくつかの研究プロジェクトをすすめている [70]。AI3 ネットワークの運用は 1996 年にはじまり、現在は日本を含む 4ヶ国のパートナーで運用を行っている。現在の AI3 ネットワークの構成を図 4.1に示す。AI3 のネットワークは Ku バンドと C バンドの静止衛星通信回線をリンクに使用し、1

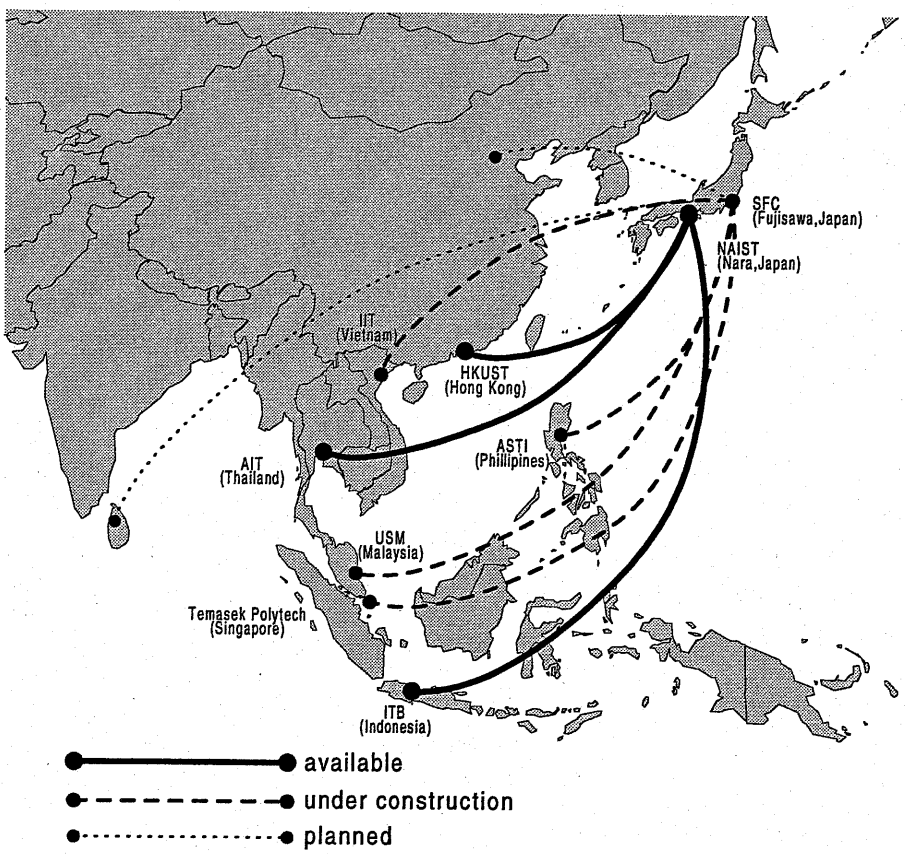


図 4.1 AI3 衛星ネットワークの構成

日 24 時間の体制で運用されている。ハブに相当する地上局は奈良先端科学技術大学院大学（NAIST: Nara Institute of Science and Technology）に設置され、リムに位置するインドネシア、香港、タイにそれぞれ双方向 1.5Mbit/s のリンクで結んでいる。現状の AI3 で使用するネットワークプロトコルは IPv4 ベースであるが、IPv6 の適用、RSVP[71]、M-Bone の運用なども行われている。また、1999 年度の終り頃に C バンドを使った 2 番目のハブ局が日本の慶應大学湘南藤沢キャンパス（SFC: Shonan Fujisawa Campus）で運用を開始する予定である。2 つのハブ局は高速の OC-3 ATM リンクで結ばれ、AI3 ネットワークにさらに数ヶ国のパートナーが加わることになる。

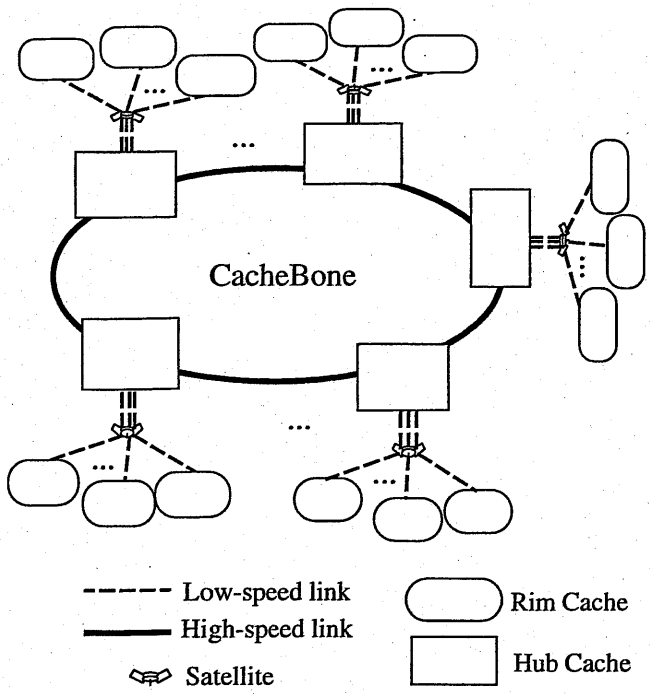


図 4.2 衛星ネットワークのキャッシュモデル

4.2.2 AI3 CacheBone

インターネットでは、複数のキャッシュサーバを協調して動作させヒット率を向上させることを目的とした国際的な試みを行っており、それらのサーバ群からなるネットワークを CacheBone と呼んでいる [46]。CacheBone ではキャッシュのヒット率を向上し、それによって利用者からみた応答時間を減少させることを目的とする。同様に、AI3 ネットワークにおける WWW キャッシュシステムを AI3 CacheBone と呼び、以下ではその適応型キャッシュシステムの設計と適用について説明を行う。

AI3 CacheBone の目的は、衛星回線におけるトラヒックを減少し、かつリム側の WWW ユーザから見た応答時間を改善することである。AI3 CacheBone では衛星ネットワークの持つスター状のトポロジを有効に活用するために、図 4.2に示すような構成をとっている。これらのキャッシュ同士の関係は階層的な親子関係あるいは隣接関係となり、複数のキャッシュが連係して動作する。リム側の複

数のキャッシュは隣接関係を持ち通常同じ組織のネットワークに置かれ高速なリンクで接続されている。サブネットワークの中心にはハブ側キャッシュが置かれ、同じサブネットワークの全てのリム側キャッシュの親キャッシュとなる。ハブとリムのキャッシュは階層的な構造となり、リム側キャッシュでキャッシュミスが発生すると、その要求は隣接キャッシュあるいはハブ側キャッシュに転送され処理される。それぞれのキャッシュ間の連係のための通信プロトコルにはICP[16]を使用している。

4.2.3 衛星ネットワークとWWWキャッシュ

ネットワークのリンクに静止衛星を使っているため、AI3ネットワークにおいて他国にアクセスする場合には衛星リンクが遅延の原因となり、またそれ自身がボトルネックになる。そこで、衛星回線のような遅延の大きい回線の影響を減らすために、いくつのキャッシュ手法が提案されている。まず、ボトルネックとなるリンクの下流側に相当する部分でWWWコンテンツのミラーリング（レプリケーション）を行うことにより、利用者からみた応答性を改善する方法がある。しかしながら、ミラーリングは一般に静的な設定で特定のコンテンツのコピーを複数の場所に定期的にコピーするようになっており、一部のコンテンツに対してのみ有効である。また、スマートキャッシュ[21]では、リンクの両端に設置した2つのキャッシュサーバが協調動作することで低速リンクがボトルネックにならないように制御を行っている。キャッシュサーバは過去のアクセスパターンを分析しそれに基づいて先読みすべきコンテンツを決定し、夜間のようにトラヒックの少ない時にコンテンツの更新と先読みを実行する。同時に、転送するコンテンツを圧縮することで使用するトラヒックを減らす方式である。しかしながら、先読みが利用者のアクセスと同時に行われないために、先読みしたコンテンツに対するヒット率が低くなるという問題がある。

AI3ネットワークの衛星リンクの帯域は1.5Mbit/sであり、比較的有効に使える帯域ではあるが、500msと大きなRTTを持っているのが問題である。WWWのアクセスに使用するHTTPプロトコルは、トランスポートのプロトコルにTCPを使用しているため、オブジェクトの取得に必要な時間はRTTに比例しまた最低でも2 RTTだけ必要である[72]。この問題を解決するために、HTTP/1.1で

導入された persistent connection と pipelining の機能がある。しかしながら、それらの機能は必要な IP パケット数や OS のソケット数を大きく減らす効果はあるが、一般に複数のオブジェクトの取得に必要な時間はかえって増加してしまう [39] という問題点がある。

先読みを使うことで、キャッシュのヒット率とオブジェクトの取得に必要な時間を短縮することができる。例えば、要求のあった WWW ページ内に存在する埋め込みオブジェクトや、他のオブジェクトへのハイパーリンクを、利用者からの要求が来る前に読み込んでおくことで利用者からみた取得時間を減少させることができる。Padmanabhan と Mogul は WWW ブラウザなどのクライアントによって先読みを行う手法を提案している [42]。この手法は他の利用者による先読みしたオブジェクトの再利用ができないという欠点がある。また、WWW のプロキシサーバにおいてページ単位で先読みを行う手法も提案されている。この方式では、利用者から要求のあった HTML で記述されたページを解析して、その中に含まれる埋め込みオブジェクトやハイパーリンク先のオブジェクトをリアルタイムで先読みし自身のキャッシュに格納する。その結果、プロキシサーバのキャッシュのヒット率は 20%高くなったが、先読みによってトラヒックが 200%増えることがわかった [43]。

衛星ネットワークにおいては、衛星回線の両側に位置する WWW クライアントとサーバ間の遅延を減少させるために、先読み技術を使用しつつ、かつ衛星回線のトラフィックを抑えることができるようなキャッシュシステムが必要とされる。

4.2.4　ハブとリムのキャッシュ

AI3 ネットワークの特徴はスター状のトポロジからなることで、その形状にに合わせて隣接キャッシュおよび親キャッシュを配置している。スター状のトポロジをもつ衛星インターネットでは、ハブ側にあたる衛星地上局に親キャッシュサーバを設置し、複数のリム側衛星地上局に子キャッシュサーバを設置するというのが有効である。その場合、ハブ側地上局に大容量かつ高速なサーバ装置を設置し、複数のリム側キャッシュの文字どおりハブとして機能する。各リム側地上局にはハブ側キャッシュと階層的な関係をもつリム側キャッシュサーバを設置する（図 4.3）。AI3 ネットワークでは、ハブ側キャッシュは日本において大容量のインター

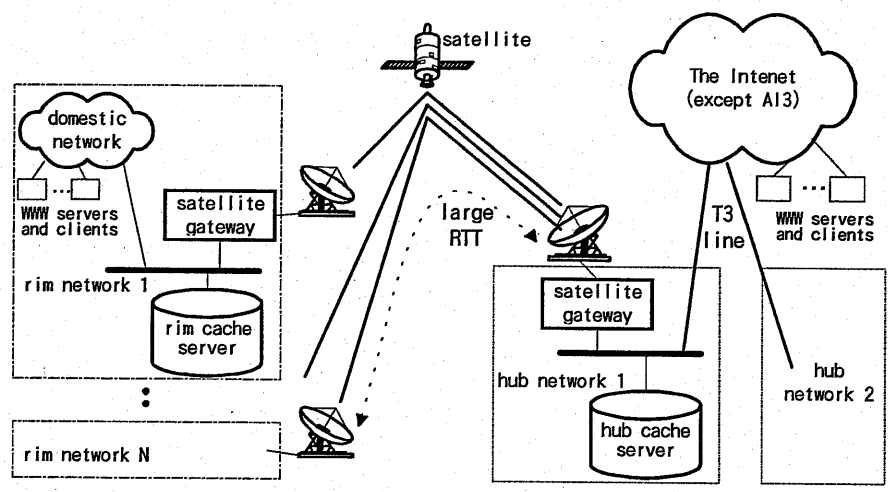


図 4.3 衛星を使ったキャッシュネットワークの構成

ネットバックボーンに接続されていることから、先読みを積極的に行いヒット率の向上と利用者の待ち時間を短縮することができる。[73]

また、キャッシュのヒット率をさらに向上させるために、ハブ側キャッシュからリム側キャッシュへWWWオブジェクトを配送する際に次の2つの方式を適用する。1つ目の方式は、ハブ側キャッシュとリム側キャッシュの間で協調動作を行わせる手法で、ハブ側キャッシュの内容を定期的に更新し、アクセスの頻度の高いWWWオブジェクトをハブ側キャッシュからリム側キャッシュへ転送すなわち複製する。各リム側キャッシュへ複製すべきオブジェクトを決定するアクセス頻度の計算には、リム側キャッシュにおけるオブジェクト（WWWページ）の統計的解析結果をもとにする。2つ目の方式は、WWWオブジェクトを複製する時にマルチキャストを使用して、複数のリム側キャッシュに同時に配送する手法である。

4.3. 衛星ネットワークのための適応型キャッシュ制御方式

4.3.1 適応型キャッシュ制御

衛星リンクの端に位置するリム側キャッシュはWWWのページを単位として、アクセスパターン分析にもとづいた先読みを行い、同時にハブ側キャッシュとの間の衛星リンクの利用率を観測し、空き帯域が存在する限り先読みを行うように制御する。基本的にリム側キャッシュはハブ側キャッシュとは独立して動作する。衛星ネットワークのための適応型キャッシュシステムの主な機能を以下の節で述べる。

4.3.2 RTTを考慮した先読み

リム側キャッシュでWWWオブジェクトを無秩序に先読みすることは、衛星リンクに大きなトラヒック負荷をかける要因となってしまう。そこで、オリジンサーバに至る経路のRTTとオブジェクトの取得に必要とした時間を考慮する方式を使用した。リム側キャッシュは稼動中に取得した各オブジェクトについて、そのオリジンサーバまでのRTTとオブジェクトの取得に要した時間を記録しておく。クライアントから要求のあったオブジェクトがその記録にあり、かつ取得に要した時間が衛星リンクのRTT（500ms）より十分に大きければ、先読みを行わない。実験で使用した閾値は20,000msとした。また、先読み使用したオブジェクトの取得時間が記録にない場合は、オリジンサーバまでのRTTから衛星リンクのRTTを差し引いたものが衛星リンクのRTTよりも大きい場合にのみ先読みを行う。

4.3.3 帯域制御

リム側キャッシュは衛星リンクの帯域消費量と遅延を観測することで、要求のあったWWWオブジェクトを先読みするかどうか決定する。衛星リンクが混雑していると、リム側キャッシュの先読みプロセスを一時的に停止し、ハブ側キャッ

シュで先読みを行う。先読みプロセスが実行される前に、衛星リンクの利用可能な帯域幅の計測を行う。利用可能な帯域幅が事前に設定しておいた閾値を下回ると、先読みのプロセスを一時的に停止する。今回の実装に使用した閾値は過去の運用上の経験値から全帯域の20%とした、すなわち帯域の80%以上が使用されていれば、先読みは行わない。

4.3.4 アクセスパターン分析を使った先読み

リム側キャッシュはクライアントからのアクセスのパターンを分析し、どのコンテンツが良く参照されるかを決定する。それをもとに、全体の遅延を最小化するためにコンテンツを先読みを行う。アクセスパターンの分析のために、WWWオブジェクト間の関係と参照される頻度の統計情報を作成する。クライアントがあるオブジェクトを要求すると、リム側キャッシュはそのオブジェクトおよび関係するオブジェクトの優先度を統計情報から決定し、その優先度に従って先読みを行う。優先度はアクセスの頻度の多いものが高くなる。

その実現方法としては、WWW のページの構成情報とリンク情報をもとにWWW オブジェクト間の関係とアクセス頻度を表すグラフを作成し、先読みを行うオブジェクトを決定する手法を使用した。リム側キャッシュでは、クライアントから要求のあったオブジェクトを、HTML ページやディレクトリなどの構造化オブジェクトと、画像やページ内の埋め込みオブジェクトなどの非構造化オブジェクトに分類する。アクセスパターン分析とそれにもとづく先読みの手順を以下に示す。

1. クライアントからのアクセスを受け付けるリム側キャッシュで、クライアントから要求のあった構造化オブジェクトの関係を表す重み付き有向グラフを作成する（図 4.4(a)）。グラフの枝の重みはリンクされたオブジェクトの参照回数を表し、クライアントがそのオブジェクトをアクセスする度に 1 ずつ値を増加させる。また、この値はエージングのために定期的に 1 ずつ減らされ、0 になった時にグラフの枝を削除する。同時に、参照される枝を持たなくなったオブジェクトはグラフから削除する。

2. クライアントがグラフ中に存在するオブジェクトをアクセスすると、キャッ

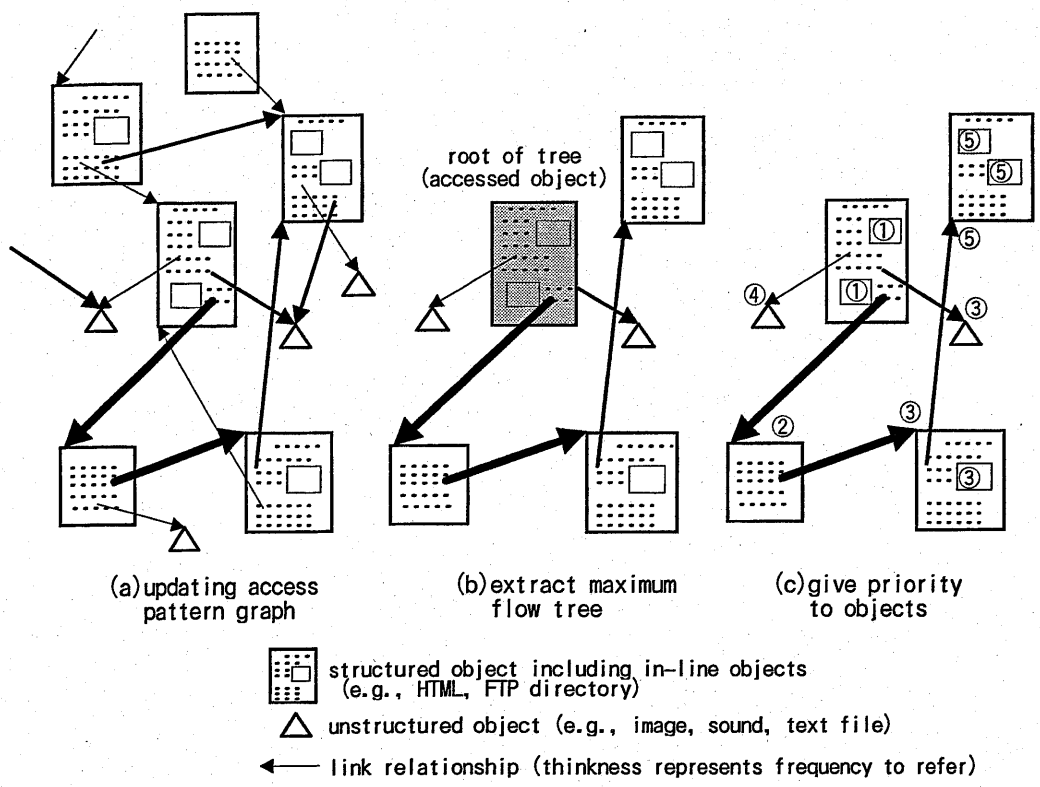


図 4.4　アクセス頻度の関係木の生成

シュサーバはそのオブジェクトを根とする有向木を抽出する（図 4.4(b)）。この有向木は元のグラフの最大フロー木を表すことになり [74]、枝の重みは根にあたるオブジェクトから参照された頻度を表す。

3. キャッシュサーバはこの木の枝の重みをもとに先読みの優先度を決める。枝の重みの大きなオブジェクトのほうが次に参照される可能性が高いと予想されるため、先読みを行う場合は枝の重みの大きなものから行う（図 4.4(c)）。

4.3.5　プッシュとマルチキャストを使用した配送

リム側キャッシュで取得した WWW オブジェクトを別のリム側キャッシュで再利用できるように、ハブ側キャッシュはインターネットからオブジェクトを取得し

てきた際に自身のキャッシュにもオブジェクトを保管し、他のリム側キャッシュからの以降の要求に備える。ここまでは通常の階層側キャッシュサーバの動作と同じである。しかしながら、衛星回線のもつ大きな遅延を解消するためには、アクセス頻度の高いオブジェクトはリム側キャッシュで保管されていることが望ましい。そこで、アクセス頻度の高いWWWオブジェクトをハブ側キャッシュから全てのリム側キャッシュに同時にマルチキャスト配送する方式を適用する。この結果、アクセス頻度の高いオブジェクトはマルチキャストによってリム側キャッシュへ配送され、それ以外のオブジェクトは階層型キャッシュシステムによって配送されることになる。なお、階層型キャッシュシステムにおいてWWWオブジェクトを効率良く配送する方法について、シミュレーションを使った研究がRodriguezらによって行われており、階層型キャッシュによる配送と更新頻度の高いものをマルチキャストを使用して配送する方式を併用して実装するのがよいと結論づけている[75]。

マルチキャストを使用した配送は十分に注意しないと意図しないネットワーク上の輻輳を引き起こす可能性がある。そこで、マルチキャスト送信における帯域消費量を制御することが必要となる。現在のネットワーク状態に従って送信レートを変化させることで、回線の帯域の効率的な利用が実現できる。提案する方式では、ある時点でネットワークでサポートできる最大レートを常に使用するように、文献[76]にあるMBFC（Monitor Based Flow Control）方式を改造したものを利用し、マルチキャストによる帯域消費量を動的に制御するようにした。MBFCは信頼性のあるマルチキャスト転送プロトコルのための送信者ベースのフロー制御方式で、TCPトラヒックのような別の種類のトラヒックが制御対象となるマルチキャストによるトラヒックと同居しているような状況を想定している。

リム側キャッシュでマルチキャストの受信状態の監視を行い、ハブ側キャッシュはその監視動作の起動また停止を制御する。各リム側キャッシュは受信結果をハブ側キャッシュに報告する。ハブ側キャッシュは受信結果を収集して、検出されたリム側キャッシュの受信状態をもとに送信レートを増減させる。ハブ側キャッシュはリム側キャッシュに一定数のマルチキャストパケットを送信し、1 RTTの間に応答が返ってくるのを待つ。その割合をもとにパケットサイズを変化させ、送信レートを調節する。変化させたパケットサイズは次に調節が必要となるまで

使用される。AI3 ネットワークで使用したマルチキャスト方式は、ネットワークの状態に応じてパケットサイズを適切な値に調節するようになっている。

4.4. 適応型キャッシュの適用と実験結果

4.4.1 リム側でのキャッシュ制御

AI3 ネットワークに適用した適応型キャッシュシステムの構成を図 4.5に示す。リム側キャッシュは 4 つのモジュールからなり、キャッシュマネージャが中心となって動作する。キャッシュマネージャはインターネットで最も広く使用されている Squid キャッシュ[12] をベースとし、他のモジュールと通信できるように改良したものを使用した。キャッシュマネージャはクライアントから HTTP で WWW オブジェクトの要求を受けとる。要求されたオブジェクトがキャッシュマネージャのキャッシュに存在すれば、そのオブジェクトをクライアントに直ちに返す (hit)。そうでなければ (miss)、キャッシュマネージャはクライアントからの要求をハブ側キャッシュに送り、その応答をクライアントに中継すると同時に自身のキャッシュに格納する。キャッシュマネージャは要求のあったオブジェクトの情報をログファイルに出力する。先読みモジュール (Prefetch) はキャッシュマネージャが出力する以下のような情報をもとに、アクセスパターン分析を行い先読みするオブジェクトを決定し先読みを行う。

- オブジェクト情報
  クライアントが要求したオブジェクトの URL、ファイルタイプ、サイズなどからなる。また、距離メトリックと呼ばれる、オリジン WWW サーバからオブジェクトを取得するのに必要であった時間も記録する。

- オブジェクトのアクセス頻度
  クライアントからある一定時間の間にアクセスのあったオブジェクトの統計的情報で、クライアントから要求があったアクセス回数からなる。

- オブジェクトの更新間隔
  あるオブジェクトがオリジン WWW サーバにおいて更新された頻度を示

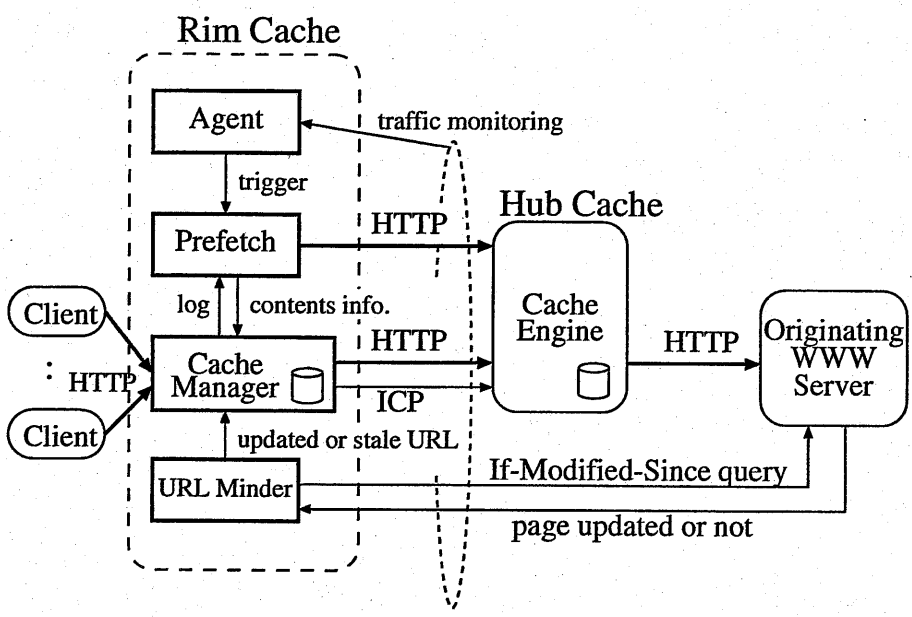


図 4.5 リムのキャッシュマネージャを中心とした構成

し、最近更新された時刻とこれまでに更新された回数からなる。

先読みモジュールはキャッシュマネージャとは独立して動作し、先読み対象のオブジェクトの取得の要求をハブ側キャッシュに対して発行する。空き帯域の検出は独立して動作するトラヒック監視モジュール（Agent）で行われ、衛星回線の空き帯域に余裕があるかどうか判断し、先読みモジュールに対して先読みの実行あるいは停止の指示を出す。先読みを行うオブジェクトの URL はそのアクセス頻度にもとづく優先度に従って先読みが実行される。優先度はキャッシュマネージャによって新しいオブジェクトが取得される度に計算される。

最も優先度の高いオブジェクトの先読みを行う際には、実際に先読みを行うかどうかの判定を行う。判定には、キャッシュマネージャから得た過去のオブジェクトの取得時間とそのオリジンサーバまでの RTT の記録を使用し、4.3.2節で述べた手順に従う。さらに、オブジェクトのサイズが判明すればそのオリジンサーバの情報とあわせて取得時間の予想が可能であり、それをもとに先読みを行うかどうかの判定も行う。先読みを行ったオブジェクトの利用率は比較的低いため [43]、リム側キャッシュでは先読みを行うオブジェクトの数をできるだけ減らすように

している。

キャッシュしたオブジェクトの一貫性を保つための仕組みを説明する。オリジンサーバにおいてオブジェクトの内容が更新された場合、現状の HTTP ではそれを検知する仕組みは特にない。オブジェクトが更新されると、キャッシュに格納されたコピーは古い内容となり、それをクライアントに返すことで一貫性の問題が生じる。この問題の解決のために、URL-minder [77] と呼ばれる仕組みを導入する。URL-minder は自動的にオブジェクトの更新を検出し通知するためのプロセスで、キャッシュの内容を定期的に検査する。オブジェクトの更新を検出を、URL-minder はキャッシュ内のオブジェクトについて HTTP の IMS（If-Modified-Since）要求をオリジンサーバに送信することで行っている。URL-minder はオブジェクトが更新されていることを検出すると、そのオブジェクトを取得すなわちキャッシュ内容を更新するようにキャッシュマネージャに指示を出す。

通常のキャッシュサーバでは、クライアントから要求されたオブジェクトを返す前に、キャッシュされた時間が設定された閾値より古い場合には、その一貫性の検査をオリジンサーバに対して行うようになっている。しかしながら、URL-minder を使用することによって、キャッシュマネージャによる同様の検査を行う必要がなくなり、クライアントに直ちにキャッシュされているオブジェクトを返す。この結果、通常のキャッシュサーバと比較して、クライアントから見た応答時間が改善されると考えられる。

4.4.2　キャッシュの性能

キャッシュの性能を評価するために、稼動状態にあるキャッシュの 1 週間の統計情報を取得した。データの収集はリム側キャッシュが設置されている Asian Institute of Technology（AIT）で、1998 年 5 月 18 日から 26 日にかけて行った。リム側キャッシュサーバは OS に FreeBSD 2.2.1 を使用し、32MB のメモリと 600MB のキャッシュディスクを持つ。

以下の性能の比較は、リム側キャッシュで行い、提案した方式にもとづくキャッシュ（adaptive Squid, 以下、適応型）と、Squid キャッシュをキャッシュマネージャと同様のパラメータで設定したもの（normal Squid, 以下、通常型）の間で行った。

4.4.2.1 ネットワークのトラヒック

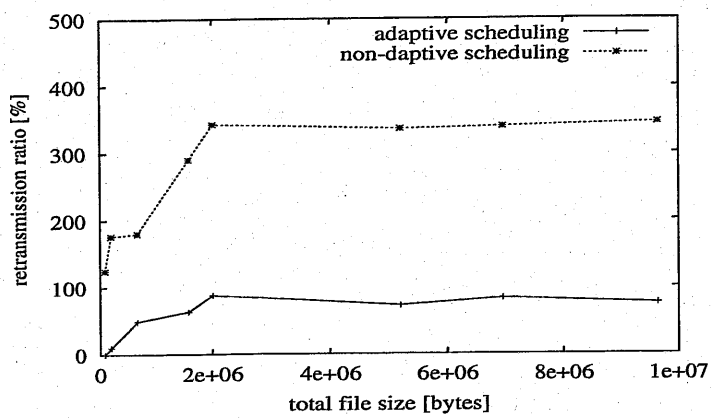
ネットワークトラヒックの観測はリム側ネットワークにおいてMRTG（Multi Router Traffic Grapher）[78]で収集したデータを利用した。MRTGは衛星ルータのインタフェースにおける通過トラヒックを収集する。5月18日から26日までの間、AI3ネットワークのAITとNAIST間のトラヒックを計測し、18日から21日までは適応型キャッシュサーバを使用し、22日から26日は通常型キャッシュサーバを使用した。帯域の最大消費量は適応型で225 kbit/s（14.65%）、通常型で131 kbit/s（8.79%）となり、適応型のほうが約7割ほど最大消費量が増加した。また、平均トラヒックは適応型が98 kbit/s（6.5%）、通常型が67 kbit/s（4.46%）となり、適応型のほうが約5割トラヒックが増加した。

適応型では先読みに起因する平均トラヒックの増加は1.5倍であるが、最大消費量の増加はより大きい。これは適応型キャッシュサーバでは利用者のリクエスト毎に先読みによるトラヒックが発生するためにバースト的な分布になっているものと考えられる。

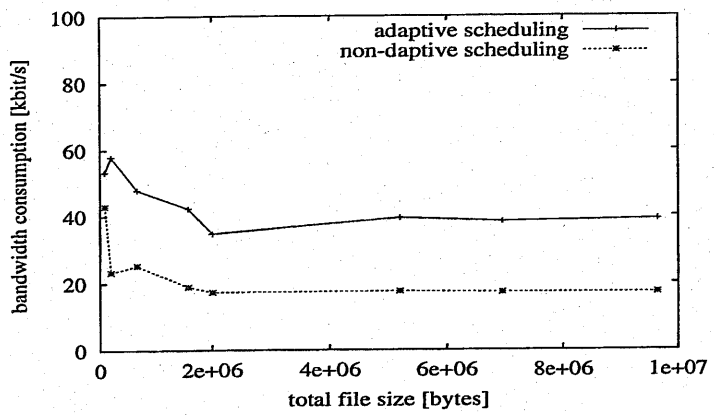
マルチキャスト送信による衛星回線のトラヒックが適応制御されているかどうか判別する指標として、ハブ側キャッシュにおけるパケットの再送率を使用する。ハブ側キャッシュは、受信側でパケット損失が発生した旨のメッセージを受信すると、それを統計情報として記録する。4.3.5節のマルチキャストによる帯域消費量を動的に制御する方式をハブ側キャッシュからのWWWオブジェクトの配送に適用した実験結果を図4.6に示す。再送率は、全転送ファイルサイズのうちに再送されたバイト数の占める割合である。帯域消費量は平均送信レートから計算した。提案した方針を使用した場合、再送率が低減することによってマルチキャスト送信の使用帯域が抑えられ、結果としてファイルサイズあたりの送信時間の短縮が実現できている。

4.4.2.2 応答時間

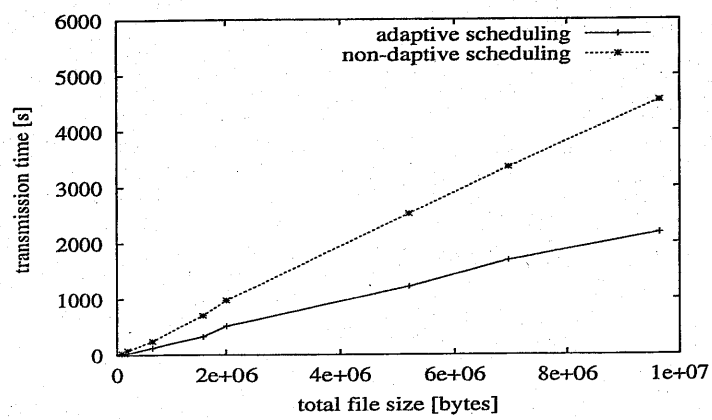
応答時間は、WWWの利用者が、あるWWWオブジェクトの要求を行ってからその取得が完了するまでの時間である。応答時間には2種類の計測方法があり、1つは利用者が使用するクライアント（ブラウザ）で観測した実際の時間で、もう1つはキャッシュサーバにおいて利用者が要求したオブジェクトを取得するの



(a) Retransmission ratio comparison



(b) Bandwidth consumption comparison



(c) Transmission time comparison

図 4.6　マルチキャストによる衛星リンクのトラヒックと配送に必要な時間

表 4.1 キャッシュヒット時とミス時の取得時間

	adaptive Squid [s]	normal Squid [s]
hit	2.53	7.3
miss	21.6	21.0

表 4.2 キャッシュヒット時とミス時のページ取得時間

object size [bytes]	adaptive Squid [s]		normal Squid [s]	
	hit	miss	hit	miss
0–10k	2.3	34.3	3.0	33.0
10–20k	2.6	63.0	3.0	65.0
20–30k	12.2	73.3	15.8	72.0
30–40k	15.0	82.5	17.0	85.0
40k–	16.0	145	18.6	144

に要した時間である。前者は取得時間を実際に計測することは困難であるが、後者はキャッシュサーバのログファイルから容易に得ることができる。クライアントとリム側キャッシュサーバは通常 LAN のような高速なネットワークで接続され、これら 2 種類の時間には大きな差はないため、ここでは後者のキャッシュサーバがオブジェクトの取得に要した時間を使用する。

適応型キャッシュと通常型キャッシュでの応答時間を計測した結果を、平均応答時間を表 4.1に、オブジェクトのサイズ毎の応答時間を表 4.2に示す。キャッシュにヒットした時は適応型で応答時間が 12%から 23%程度短縮されており、ミスした時はほぼ同じ時間となっている。サイズ分布の大部分を占める 0–10kbytes においては、約 2 割短縮されている。この理由として、適応型では URL-minder によって WWW オブジェクトのオリジンサーバとの一貫性が保たれているため、余

表 4.3 適応型と通常型のキャッシュのヒット率

	adaptive Squid	normal Squid
request hit ratio [%]	26.3	10.6
bytes hit ratio [%]	16.5	4.8

計な IMS 要求を発行し結果を待つ時間が不要となったためと考えられる。現在の実装では、URL-minder による IMS 要求はキャッシュ内の各オブジェクトについて一日一回の割合で発行するようしている。

4.4.2.3　ヒット率

キャッシュのヒット率には 2 種類のものが考えられる。1 つは要求ヒット率（request hit ratio）で、キャッシュサーバが受けとった要求のうちキャッシュにヒットした要求数の占める割合で、通常キャッシュのヒット率といえばこちらを指す。他方はバイトヒット率（byte hit ratio）で、キャッシュサーバが受けとった要求の総バイト数のうちキャッシュにヒットした要求の総バイト数の占める割合で、トラヒックの解析などに有用である。なお、キャッシュのヒット率はキャッシュサーバのログファイルを解析して算出したものである。

表 4.3に、適応型と通常型のキャッシュのヒット率を示す。ヒット率は比較的低いように見えるが、同時期の他のグローバルなキャッシュと比較してそれほど低いわけではない。たとえば、NECTEC に設置されているキャッシュの 1998 年 5 月のヒット率は約 13.2% [79] であり、また NLANR で 1998 年 5 月のヒット率は約 29% [80] であった。この表に示すように、適応型における要求ヒット率は通常型と比較して約 15%増加し、バイトヒット率は約 12%増加した。

4.4.3　キャッシュの自動構成管理

前章で論じたキャッシュの自動構成管理を AI3 のキャッシュシステムに適用する。リム側キャッシュにおける構成情報を親および隣接キャッシュに依存しない記

```
#           hostname      type        HTTP/ICP port

cache_host  local_cache1  sibling     8080 3130
cache_host  local_cache2  sibling     8000 8130
cache_host  hub_cache1    parent      3128 3130
cache_host  hub_cache2    parent      3128 3130
cache_host  hub_cache3    parent      3128 3130
```

(a) Configuration File before Introducing HintServer

```
#           hostname      type        HTTP/ICP port

cache_host  local_hints1  hintserver  -    4649
cache_host  hub_hints1    hintserver  -    4649
cache_host  hub_hints2    hintserver  -    4649
```

(b) Configuration File after Introducing HintServer

図 4.7 ヒントサーバの導入前と後でのリム側キャッシュの構成定義の比較

述にすることができるため、キャッシュの構成管理の自動化・最適化が可能となる。ヒントサーバを導入する前後での、リム側キャッシュにおける構成定義の内容を比較する。あるリム側キャッシュでは2台のローカルキャッシュサーバと3台のハブ側キャッシュを参照していたとすると、その構成定義の記述は図4.7(a)のようになる。この状態で、ローカルにヒントサーバを1台、ハブ側にヒントサーバを2台設置した場合の構成定義の記述は図4.7(b)のようになる。このようにヒントサーバをシステムに導入することで、親キャッシュサーバの増減、設定変更に自動的に対応できるようになる。

4.5. むすび

キャッシュ技術はWWWサービスにおけるサービス品質を改善するために効果がある。提案した衛星ネットワークのための適応型キャッシュでは、トラヒック監視やオブジェクトの存在するサーバまでのRTTなどをもとに、キャッシュ内容の先読みを制御することで、ヒット率の改善とオブジェクトの取得時間の改善を実現できた。さらに、プッシュ技術とマルチキャストを利用することでオブジェ

クトの再配布を実現し、従来システムに比べて大きく使用帯域が増えないことを確認した。また、キャッシュ内のオブジェクトの更新のために URL-minder を導入したが、その更新間隔は本来のオブジェクトの更新間隔に比例しておらず、この値を最適化していく必要がある。また、新しいハブ局の運用が開始される際にはヒントサーバを使った構成自動化の検証を行なう必要がある。

第 5 章　結論

本章では、本研究で得られた成果をまとめ、今後の課題について述べ、インターネットの今後とWWWキャッシュシステムの展望について議論する。

5.1.　本研究で得られた成果

5.1.1　分散キャッシュの自律構成における問題点とその解決

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。

5.1.2　衛星ネットワークにおけるWWWキャッシュ制御方式

提案した適応型キャッシュでは、トラヒック監視やオブジェクトの存在するサーバまでのRTTなどもとに、キャッシュ内容の先読みを制御することで、ヒット率の改善とオブジェクトの取得時間の改善を実現できた。さらに、プッシュ技術とマルチキャストを利用することでオブジェクトの再配布を実現し、従来システムに比べて大きく使用帯域が増えないことを確認した。

5.2. 今後の課題

5.2.1 ヒントサーバを使ったモデルの問題点と検証すべき事項

ヒントサーバは、各キャッシュサーバからキャッシュ内容の変更通知を受けとることにより、ヒントサーバのデータベースとキャッシュの内容の整合性を取る。しかしながら、変更通知が届かないなどの障害が起り、この整合性が取れなくなるとヒントサーバはキャッシュサーバに誤った情報を渡してしまうことになるため、この問題がどの程度ヒット率や応答時間に影響を及ぼすかを確認していく必要がある。

また、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実装していく必要がある。

5.2.2 ハブキャッシュが複数ある場合の適用実験

今回実験を行った、AI3衛星ネットワークはハブとなるキャッシュが1箇所であったが、2000年初頭には日本側の衛星ハブ局が新設され2箇所となり、またリム側の拠点も増加する予定である。これに合わせて、提案した方式の適用範囲を広げ検証していく必要がある。また、ヒントサーバを使った構成管理手法をAI3ネットワークにも適用し評価する必要がある。

なお、実験を通じて、AI3のように文化的・言語的に異なる国際間のネットワークでは、リムごとのWWWアクセスの傾向が大きく異なるため、コンテンツの共有は効率が悪い可能性があるらしいことがわかった。この点はまだ定性的に評価できていないが、今後リムのキャッシュ間のアクセス内容の類似性などに着目していく必要がある。

5.3. インターネットの今後とWWWキャッシュシステムの展望

この5年間におけるインターネットの普及とトラヒックの増加はめざましいものがある。今後もWWWを中心とした利用は当分続くと考えられ、WWWキャッシュシステムは今後とも有効であろう。なお、現在の伸び率ではトラヒックは半年から1年で倍以上になってしまうため、トラヒックの削減を目的としたWWWキャッシュシステムには意味がないのではないかという議論もある。例えば、WWWキャッシュを適用して平均30～40%のヒット率があればトラヒックは約3分の2になる。しかしながら、この展開の早いインターネットの世界で、半年から1年の間、回線や設備の増強を猶予できるというのは十分意義がある。

インターネットの発達の速さの仕組みを考えてみると、図5.1に示すように、インターネットでのサービスの充実・拡大が、利用者の増加につながり、またインフラの整備がすすむ。そして、投資および新ビジネスの対象となり、サービスの充実につながる。このように、4つの要因が相互に作用することで、現在のような加速度的な発展になっている。なかでも、オンラインショッピング、オンライントレード、企業間の購買へのインターネットの利用、また、VoIP (Voice over IP) などを使ったIP電話の普及、IPv6による家電製品や自動車などの移動体がオンラインとなること、さらに、携帯電話の爆発的普及と携帯端末としての利用や個人利用者のインターネット常時接続の広がりから、インターネットの利用度とそのトラヒックはさらに増え続けるであろう。

今後のWWWキャッシュの役割としては、コンテンツ配送（ディストリビューション）システムの一部として、より効果的な配送を行うための構成要素として活用されていくべきであると考えている。たとえば、シンクライアント（thin-client: アプリケーションデータを持たずネットワーク経由で情報を入手するタイプのクライアント装置で、WWWブラウザを中心としたミドルウェアからなり、維持コストが低いという特徴を持つ）と、その情報を提供するためのASP（Application Service Provider）による効果的なネットワークコンピューティングを実現するためには、情報の配送システムを確立する必要がある。

この数年間のインターネットをめぐる動きは非常に激しいものがあるが、基本

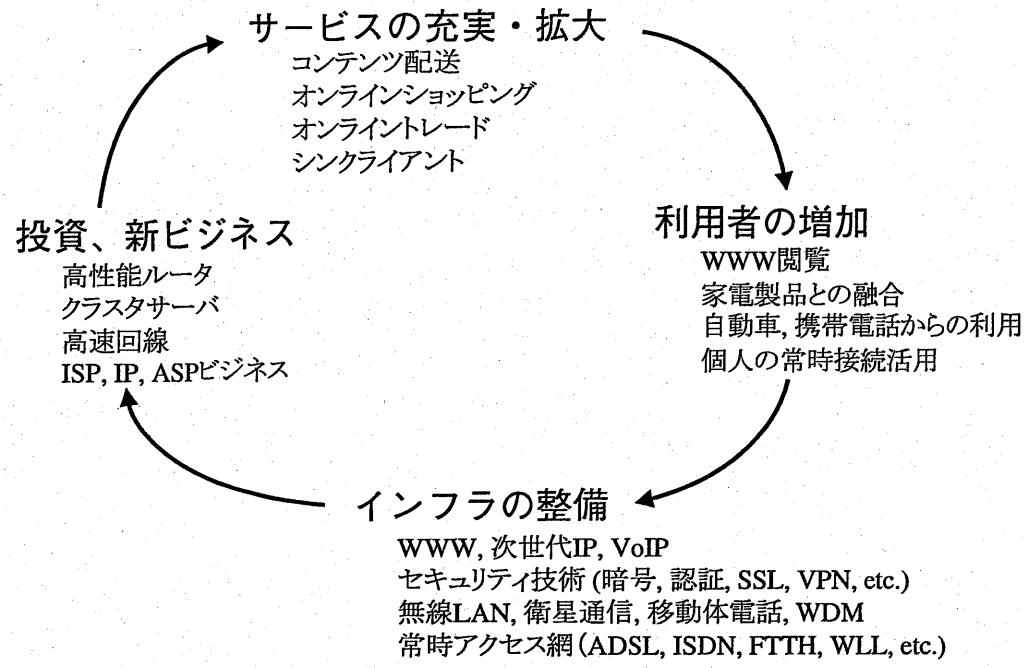


図 5.1 インターネットの発展の相互作用

的な IP プロトコルが動き出してから実に 20 年以上が経過している。その中では、WWW という技術はまだ新しいものであり、今後 10 年、20 年のちに、あるべきシステムを目指して引続き積極的にこの分野に関与していきたい。また、WWW キャッシュシステムは、プロトコル技術、分散処理、ネットワーク構築、運用などの広い分野からなっており、今後も幅広い視点からネットワークシステムを研究していきたい。

謝辞

本研究の全過程を通じて、終始懇切な御指導、御鞭撻を賜わりました奈良先端科学技術大学院大学情報科学研究科の山本平一教授、九州工業大学情報工学部の尾家祐二教授、終始一貫して御指導いただいた奈良先端科学技術大学院大学情報科学研究科の山口英助教授に心よりの感謝の意を表します。また、貴重な御助言と御指導を賜わりました奈良先端科学技術大学院大学情報科学研究科の千原國宏教授、福田晃教授に深く感謝の意を表します。

本研究に関する議論やアドバイス、実験設備の提供、また、研究を遂行するうえで必要な作業などをいろいろと御支援くださった、奈良先端科学技術大学院大学情報科学研究科情報ネットワーク講座の知念賢一助手、谷田奈津江氏、出水法俊氏、森島直人氏、木村泰司氏、大江将史氏、像情報処理学講座の楳田敏之氏、同大情報科学センターの砂原秀樹助教授、同大附属図書館研究開発室の羽田久一助手、倉敷芸術科学大学芸術学部美術学科の馬場始三講師、同大学産業科学技術学部ソフトウエア学科の小林和真助教授、岡山大学工学部情報工学科の岡山聖彦助手、そして大阪大学大型計算機センターの門林雄基講師に感謝の意を表します。

ヒントサーバのアイデアは、WIDE プロジェクトの W4C ワーキンググループでの議論の中から生まれたものです。議論やアドバイスをいただいた W4C ワーキンググループの奈良県工業技術センターの坂本佳則氏、日立製作所の吉田健一氏、松井証券の民田雅人氏、北陸先端科学技術大学院大学の篠田陽一助教授、松下電気産業の坂本岳史氏、ならびにメンバの諸氏に感謝の意を表します。AI3 の衛星ネットワークにおける議論、運用、実験などに御協力いただいた Asian Institute of Technology の Kanchana Kanchanasut 助教授、Kriengsak Kiatsirivatana 氏、Panjai Tantatsanawong 氏、そして日本サテライトシステムズの諸氏に感謝の意を表します。

本研究において様々な御協力と御支援、貴重なアドバイスを頂きました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の上原明所長、村瀬亨部長、田口哲也主任研究員とそのメンバの方々、同社の内田恒裕取締役、同社情報通信システム事業部の高橋秀公次長に心より感謝の意を表します。

本研究に関して様々な分野での議論を行なってくださった、奈良先端科学技術大学院大学情報科学センターおよび同大情報科学研究科情報ネットワーク講座の卒業生を含む学生のみなさま、また WIDE プロジェクトの諸氏に感謝の意を表します。

最後に、今日まで筆者の研究活動に対する理解と協力をいただいた妻 美和と長男 龍一に感謝の言葉をおくります。

参考文献

[1] 横河デジタルコンピュータ (株)SI 事業本部（高橋徹）："インターネット商用化に向けて（CIX）", トッパン, Nov. 1993.

[2] 出水法俊："インターネット最前線 – World Wide Web Consortium –", bit, vol. 28, no. 9, pp. 79–86, Sept. 1996.

[3] ロバート・リード："インターネット激動の 1000 日間", 日経 BP, May 1997.

[4] Nielsen//NetRatings: "インターネット人口調査概要 (1999 年 11 − 12 月)", http://www.netratings.co.jp/press_releases/pr_010200.htm, Feb. 2000.

[5] 郵政省："通信白書 平成 11 年版", July 1999.

[6] Internet Software Consortium: "Internet Domain Survey", http://www.isc.org/ds/, July 1999.

[7] WIDE Project: "WIDE / NSPIXP Home Page", http://jungle.sfc.wide.ad.jp/NSPIXP/.

[8] M. Abrams and et al.: "Caching Proxies: Limitations and Potentials", Proceedings of the 4th WWW Conference, pp. 119–133, 1995.

[9] T. Berners-Lee, L. Masinter and M. McCahill: "Uniform Resource Locators (URL)", RFC 1738, Dec. 1994.

[10] The World Wide Web Consortium: "CERN httpd (W3C httpd)", http://www.w3.org/pub/WWW/Daemon/, 1996.

[11] A. Chankhunthod and et al.: "A Hierarchical Internet Object Cache", Technical Report 95-611, Computer Science Dep., Univ. of Southern California, 1995.

[12] National Laboratory for Applied Network Research: "Squid Internet Object Cache", http://squid.nlanr.net/.

[13] M. Reardon: "Caches Keep Content Close at Hand", Data Communications, pp. 47–55, Mar. 1999.

[14] A. Rousskov, D. Wessels, G. Chisholm and D. Newman: "Proxy Caches: Speeding Up the Web", Data Communications, pp. 57–70, Oct. 1999.

[15] C. Schimmel: "UNIX カーネル内部解析 – キャッシュとマルチプロセッサの管理", ソフトバンク, Apr. 1996.

[16] D. Wessels and K. Claffy: "Internet Cache Protocol (ICP), version 2", RFC 2186, Sept. 1997.

[17] WIDE プロジェクト："WWW キャッシュ技術", WIDE プロジェクト研究報告書 1998, pp. 501–548, 1999.

[18] 鍋島公章："キャッシュサーバ運用技術", Internet Week '99 チュートリアル, Dec. 1999.

[19] P. Gauthier, J. Cohen, M. Dunsmuir and C. Perkins: "Web Proxy Auto-Discovery Protocol (WPAD)", IETF Draft, draft-ietf-wrec-wpad-01.txt, July 1999.

[20] National Laboratory for Applied Network Research: "Cache Registrations", http://ircache.nlanr.net/, 1996.

[21] G. V. Dias, G. Cope and R. Wijayaratne: "A Smart Internet Caching System", Proceedings of INET'96, June 1996.

[22] "The Problems of Offshore Content and Caching", CDA フィラデルフィア連邦地裁判決 118, 民事訴訟 96 年 1458 号, June 1996.

[23] US Government: "Digital Millennium Copyright Act of 1998", Oct. 1998.

[24] 牧野二郎："キャッシュの法律問題 – 主に著作権「複製権」との関連に関する考察–", http://www3.justnet.ne.jp/~ilc/journal/990429_1.htm, Apr. 1999.

[25] Real Networks: "Real.com", http://www.real.com/.

[26] Cisco Systems: "Cisco IP/TV Streaming Video", http://www.cisco.com/warp/public/cc/cisco/mkt/video/iptv/.

[27] Akamai Technologies, Inc.: "Akamai", http://www.akamai.com/.

[28] A. Gulbrandsen and P. Vixie: "A DNS RR for specifying the location of services (DNS SRV)", RFC 2052, Oct. 1996.

[29] K. W. Ross: "Hash Routing for Collections of Shared Web Caches", IEEE Network, vol. Nov./Dec. 1997, pp. 37–44, Jan. 1998.

[30] Microsoft Corporation: "Cache Array Routing Protocol and Microsoft Proxy Server 2.0 – White Paper", http://www.microsoft.com/ISN/isp/whitepapers.asp, Aug. 1998.

[31] Standard Performance Evaluation Corporation: "SPECweb99", http://www.spec.org/.

[32] IRCache Project: "Web Polygraph – proxy performance benchmark", http://www.ircache.net/Polygraph/.

[33] IRCache Project: "UCSD/IRCACHE Bake-offs – caching products competition", http://bakeoff.ircache.net/.

[34] W. R. Cheswick and S. M. Bellovin: "Firewalls and Internet Security", Addison-Wesley, Apr. 1994.

[35] D. B. Chapman and E. D. Zwicky: "Building Internet Firewalls", O'Reilly & Associates, Sept. 1995.

[36] 佐藤豊："インターネット防火壁の基礎技術と応用 – DeleGate の仕組み –", コンピュータソフトウェア, vol. 14, no. 1, 1997. (http://www.delegate.org/delegate/).

[37] T. Berners-Lee, R. Fielding and H. Frystyk: "Hypertext Transfer Protocol – HTTP/1.0", RFC 1945, May 1996.

[38] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee: "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616, June 1999.

[39] H. F. Nielsen, J. Gettys, A. Baird-Smith and E. Prud'hommeaux: "Network Performance Effects of HTTP/1.1, CSS1, and PNG", http://www.w3.org/pub/WWW/Protocols/HTTP/Performance/Pipeline.html, Feb. 1997.

[40] A. Luotonen and K. Altis: "World-wide Web Proxies", Proceedings of First International World-Wide Web Conference, Apr. 1994.

[41] 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", 電子情報通信学会論文誌 D-I, vol. J80-D-I, no. 11, pp. 907–915, Nov. 1997.

[42] V. N. Padmanabhan and J. C. Mogul: "Using Predictive Prefetching to Improve World Wide Web Latency", ACM SIGCOMM Computer Communication Review, July 1996.

[43] K. Chinen and S. Yamaguchi: "An Interactive Prefetching Proxy Server for Improvements of WWW Latency", Proceedings of INET'97, June 1997.

[44] N. J. Yeager and R. E. McGrath: "Web Server Technology", Morgan Kaufmann, 1996.

[45] M. Baentsch, L. Baum, G. Molter, S. Rothkugel and P. Sturm: "Enhancing The Web's Infrastructure: From Caching To Replication", IEEE Internet Computing, vol. March/April, pp. 18–27, Apr. 1997.

[46] "the Cache Now! campaign", http://vancouver-webpages.com/CacheNow/.

[47] D. Wessels and K. Claffy: "Application of Internet Cache Protocol (ICP), version 2", RFC 2187, Sept. 1997.

[48] Network Appliance, Inc.: "NetCache appliances", http://www.netapp.com/products/netcache/.

[49] 内藤広志："Squidによる分散WWW Cacheの有効利用法", 日本UNIXユーザ会'97大規模ネットワーク管理シンポジウム, 1997.

[50] Netscape Corporation: "Navigator Proxy Auto-Config File Format", http://www.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html, 1996.

[51] E. Guttman, C. Perkins, J. Veizades and M. Day: "Service Location Protocol, Version 2", RFC 2608, June 1999.

[52] E. Guttman, C. Perkins and J. Kempf: "Service Templates and Service: Schemes", RFC 2609, June 1999.

[53] S. Deering: "Host extensions for IP multicasting", RFC 1112, Aug. 1989.

[54] J. Postel: "DoD statement on the NRC report", RFC 945, May 1985.

[55] シャープ (株)："URL ハッシュ式分散 Proxy キャッシュ", http://naragw.sharp.co.jp/sps/index.html, 1996.

[56] A. Rousskov and D. Wessels: "Cache Digests", Proceedings of 3rd International WWW Caching Workshop, June 1998.

[57] 酒井明広："WWWにおけるストリーム制御に基づいた通信量削減技術に関する研究", 修士論文, 奈良先端科学技術大学院大学, Mar. 1998.

[58] Cisco Systems: "Cisco Cache Engine User Guide – Web Cache Communication Protocol", http://www.cisco.com/univercd/cc/td/doc/product/iaabu/webcache/ce20/ver20/, 1999.

[59] Cisco Systems: "Shared Network Caching and Cisco's Cache Engine", White Paper, 1997.

[60] IBM Corporation: "Load Balancing Among Internet Servers at UCLA", http://www.csc.ibm.com/advisor/provensolutions/pcid/75a6_602.html, Aug. 1996.

[61] 多田信彦, 山口英, 山本平一："Service Switching 技術の研究", 情報処理学会研究報告 94-DPS-65, Apr. 1994.

[62] 馬場始三, 篠田陽一："第 1 回インターネット防災訓練における生存者情報データベース", インターネットコンファレンス'96, July 1996.

[63] 馬場始三, 山口英, 尾家祐二："DNS を用いた広域負荷分散の実装", DSM 研究会 98 年度第一回定例研究会, May 1998.

[64] T. Asaka, H. Miwa and Y. Tanaka: "Hash-Based Query Caching Method for Distributed Web Caching in Wide Area Networks", IEICE Transactions, vol. E82-B, no. 6, pp. 907–914, June 1996.

[65] P. Srisuresh and M. Holdrege: "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, Aug. 1999.

[66] P. Srisuresh and D. Gan: "Load Sharing using IP Network Address Translation (LSNAT)", RFC 2391, Aug. 1998.

[67] 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, vol. 97-DPS-85, pp. 201–206, Nov. 1997.

[68] J. L. Hennessy and D. A. Patterson: "Computer Architecture A Quantitative Approach (2nd ed.)", Morgan Kaufmann, 1996.

[69] D. Chaiken, C. Fields, K. Kurihara and A. Agarwal: "Directory-Based Cache Coherence in Large-Scale Multiprocessors", Computer, vol. 23, no. 6, pp. 49–58, 1990.

[70] S. Yamaguchi and J. Murai: "Asian Internet Interconnection Initiatives", Proceedings of INET'96, June 1996.

[71] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog and S. Jamin: "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", RFC 2205, Sept. 1997.

[72] J. C. Mogul: "The case for persistent-connection HTTP", Proceedings of ACM SIGCOMM'95, Oct. 1995.

[73] K. Kanchanasut, S. Yamaguchi, K. Kiatsirivatana, H. Inoue, A. Suksakorn and P. Tantatsanawong: "The AI3 CacheBone Project", Proceedings of IWS'99, pp. 259–264, Feb. 1999.

[74] 伊理正夫, 白川功, 梶谷洋司, 篠田庄司："演習グラフ理論", コロナ社, Apr. 1983.

[75] P. Rodriguez, W. E. Biersack and W. K. Ross: "Improving the WWW: caching or multicast", Proceedings of Third International WWW Caching Workshop and TERENA TF-Cache Meeting, June 1998.

[76] T. Sano, T. Shiroshita, O. Takahashi and N. Yamanouchi: "Flow and congestion control for bulk reliable multicast protocols - toward coexistence with TCP", RMTP Publications, IBM Tokyo Research Laboratory, 1998.

[77] "Netmind", http://www.netmind.com/html/deploy.html, May 1998.

[78] "MRTG: The Multi Router Traffic Grapher", http://www.ee.thhz.ch/~oetiker/webtools/mrtg/mrtg.html, May 1998.

[79] "NECTEC Cache Server Monthly Statistics Report", http://ntl.nectec.or.th/pubnet/services/cache/monthly.html, May 1998.

[80] "NLANR Cache Server Daily Statistics Report", http://ircache.nlanr.net/Cache/Statistics/Reports/bo1.cache.nlanr.net/199805/report.19980524, May 1998.

著者研究業績

1. 学術論文誌

(1) 井上博之, 坂本岳史, 山口英, 尾家祐二："分散WWWキャッシュシステムの構成自動設定機構", 情報処理学会論文誌, vol. 40, no. 12, Dec. 1999.

(2) 井上博之, Kanchana Kanchanasut, 馬場始三, 山口英："AI3衛星ネットワークにおけるWWWキャッシュ制御方式", システム制御情報学会論文誌. [投稿中]

2. 国際会議（査読つき）

(1) H. Inoue, K. Kanchanasut and S. Yamaguchi: "An Adaptive WWW Cache Mechanism in the AI3 Network", Proceedings of INET'97, June 1997.

(2) H. Inoue, T. Sakamoto, S. Yamaguchi and Y. Oie: "WebHint: Automatic Configuration Mechanism for Optimizing World Wide Web Cache System Utilization", Proceedings of INET'98, July 1998.

3. 研究会

(1) 井上博之, 山口英："NATによるWWWサーバの負荷分散機構の実装", 情報処理学会研究報告 マルチメディアと分散処理研究会, 96-DPS-78, pp.19–24, Sep. 1996.

(2) 井上博之, 村瀬亨：“公衆網向け ADSL システムの構成の検討”, 1998 信学会ソサイエティ大会, B-7-6, Sep. 1998.