マルチメディア通信と分散処理 74－28
グループウェア 15－28
（1996. 1. 26）

広域ネットワーク環境における情報分散メカニズムの一検討

小口正人、小野欽司
学術情報センター 研究開発部
{oguchi,ono}@rd.nacsis.ac.jp

World-Wide Web(WWW) に代表される広域分散ネットワーク上での情報検索・収集システムの利用は、爆発的な広がりをみせている。これらのシステムにおいて、アクセス性の向上などの面でキャッシュリレーは非常に大きな役割を持っている。

本稿ではこのキャッシュリレーの新しい実現方式について検討する。まずキャッシュリレーの概念と仕組みを紹介した後、現行のキャッシュリレーの問題点を指摘する。次にこれらの問題点に対処する方法として、品質レベルのコントロールによるファイルサイズの変更をキャッシュリレーで行うこと、および近傍のキャッシュリレー間でクラスタを形成しハイパーリンク情報を用いて連携を行うことを提案する。そしてこれらの方針を組み合わせたキャッシュリレーメカニズムの実現方式を示す。

A Study on Information Dispersal Mechanism in a Wide Area Network Environment

Masato OGUCHI and Kinji ONO
Research and Development Department
NACSIS (National Center for Science Information Systems)

The information retrieval systems on a wide area distributed network, such as World-Wide Web(WWW), become popular among the extremely large number of users. The caching relay has an important role on these systems for improving accessibility and serviceability.

In this paper, the caching relay mechanism is discussed from following aspects. First, the role and structure of the caching relay are explained, and problems of the existing systems are pointed out. Next, two measures are proposed in order to overcome the problems of the existing systems. An improved caching relay mechanism, based on these ideas, is shown finally.

1 はじめに

World-Wide Web(WWW)[1]に代表される広域分散ネットワーク上での情報検索・収集システムの利用は、爆発的な広がりをみせている。この理由として一つには、これらのシステムは完全な分散システムであり、ユーザが気軽に情報を収集できるだけでなく、誰に管理されることもなく気軽に情報を発信できる点があげられる。また特にWWWなどにおいては、テキストだけでなく静止画、動画、音声といったマルチメディアデータも容易に扱うことができ、このことがユーザにとって大きな魅力となっている。

しかし一方でこれらのシステムは、もともと均一性の非常に低い広域分散ネットワークであるInternet上に構成されており、全体のパフォーマンスなどを綿密に検討された上で設計されたシステムではないために、当然さまざまな問題点が存在する。例えば地理的に遠い、あるいは回線が細いなどの理由により接続性の悪いサイトから動画ファイルのような大きなデータをダウンロードしようとすると、とても実用的でない程長い時間がかかることがある。またユーザの関心が高いサイトへは非常に多くのアクセスが集中し、これによりサーバ側が対応しきれなくなったり、サービス性が悪くなったりすることがしばしば起こる。

このような問題点に対処する仕組みの一つとして、キャッシュリレーが提案され、広く用いられるようになってきた。本稿ではこのキャッシュリレーの新しい実現方式について検討する。まずキャッシュリレーの概念と仕組みを紹介した後、現行のキャッシュリレーの問題点を指摘する。そしてこれらの問題点に対処する方法を提案し、これに基づいたキャッシュリレーメカニズムの実現方式について述べる。

2 ネットワーク上での情報収集システムにおけるキャッシュリレー

WWWに代表されるネットワーク上での情報検索・収集システムにおいて、キャッシュリレーは非常に大きな役割を持っている。キャッシュリレーは、Proxyサーバ上に実現されたファイル単位でのキャッシュシステムである[2]。

Proxyサーバはクライアントからリソースサーバへのリクエストを仲介するシステムで、特にファイアウォールを導入したサイトにおいて外のサイトとのやり取りを行う必要性が存在することから広く用いられるようになった。クライアントはリクエストを、直接リソースサーバに送る代わりにProxyサーバへ送る。Proxyサーバはリソースサーバとやり取りを行って必要なファイルをダウンロードし、これをクライアントへ届ける。この際にProxyサーバがファイルを自分のディスクにもストアしておき、次に同じファイルへのリクエストが来た時にはリソースサーバへアクセスせずに自分のディスクの中から取り出してクライアントへ返す、というのがキャッシュリレーの基本的な動作である。

キャッシュリレーの概念図を図1に示す。キャッシュリレーはInternetのような広域分散ネットワークにおいては非常に有効である。ローカルな環境にキャッシュリレーを導入することにより、キャッシュがヒットした場合、地理的にあるいは回線の太さの面で遠いサイトへのアクセスが、ローカルなサイトへのアクセスと同等になるからである。これによりユーザのリクエストデータへのアクセス時間を短縮できるだけでなく、ネットワーク上を流れるトラヒックを押える効果も期待できるので、広域ネットワーク環境においては非常に好ましいと考えられる。

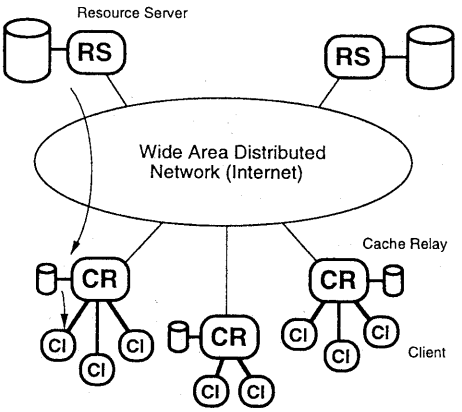

図 1: An overview of cache relay

一方情報収集システムをサーバ側の視点から見た場合、広域ネットワーク上のユーザの数は非常に多いだけに、ユーザの関心が高いサイトへのアクセス集中はかなり深刻な問題となっている。キャッシュリレーは、このようなサーバに対するアクセス集中を緩和する効果も持っている。アクセス集中の問題に関しては、同時に多くのリクエストに応えることができるようリソースサーバの構成を並列・分散化することなどが検討されている[3, 4]。これに対しキャッシュリレーはサーバへのアクセス数そのものを低減する働きがあり、サーバ自身の並列・分散化と組み合わされてアクセス集中の緩和に大きな効果があると期待される。

またネットワーク上の情報資源へのアクセスパターンには局所性が存在し、キャッシュのヒット率は割合高いことが知られている。例えばある統計によると、一日に60人のユーザから2000リクエストがあるような大きなサイトの場合キャッシュヒット率は30-50%、20人のユーザから数百程度のリクエストがあるようなやや小さいサイトでもヒット率は25-35%と報告されている[5]。

3 キャッシュリレーにおける問題点

前節で述べたように、広域分散ネットワーク上の情報収集システムにおいてキャッシュリレーの果たす役割は非

常に大きい。しかしキャッシュリレーにもいくつかの問題点が存在する。現行のシステムにおいて、主に以下の二点が大きな問題であると考えられる。

1. マルチメディアデータのような大きなサイズのファイルをキャッシュすることは困難である
2. 近傍のキャッシュリレー間の連携がないため、すぐ近くのキャッシュリレーにリクエストファイルが存在してもこれを活用することができない

3.1 ファイルサイズの問題

ファイルサイズの問題については、近年マルチメディア技術が広がり、ネットワーク上でも例えばMPEGの動画ファイルのように非常に大きなファイルがやり取りされるようになったことにより、より深刻なものとなっている。プログラムのソースコードなどは同じサイトのユーザが何度も繰り返しアクセスしてダウンロードするようなことは稀であるが、動画や音声などのマルチメディアファイルは繰り返しアクセスされる可能性が高いからである。

一般にキャッシュリレーは、非常に大きいサイズのファイルをキャッシュすると他のファイルのスペースが圧迫されてしまうため、ある大きさ以上のファイルの場合にはこれをキャッシュせず、リクエストの仲介だけを行う。この上限の大きさはパラメータとして設定することができ、管理方針により自由に変更できるのであるが、キャッシュリレー全体の効率を考えて一般に数百 KB～数 MB 程度に設定される。従ってテキストファイルやJPEG、GIFなどの静止画像ファイルは、多くの場合キャッシュされる。しかし動画象ファイルや高品質なオーディオファイルなどの場合には、数 MB～数十 MB 以上となるものも多いため、これらは一般にキャッシュされないことが多い。これらのファイルはダウンロードするのに時間、トラヒック両面で大きなコストを要するにもかかわらず、キャッシュされないため、再びアクセスされるとまた大きなコストをかけて元のサイトからダウンロードしなおさなければならない。

3.2 キャッシュリレー間の連携

Proxy サーバはそれ自体が他の Proxy サーバからのリクエストを受け付けることが出来るため、Proxy サーバを階層構造に構成することは可能である。しかし現実的にはキャッシュ効果が発揮できるのは二階層程度までであり [5]、多くの場合は階層構造を取っていない。そして同じレベルのキャッシュリレー間の、すなわち水平方向の連携は存在しない。

このことにより、例えばあるキャッシュリレーにおいてクライアントからリクエストを受けたファイルがすぐ近くの別のキャッシュリレーに存在していたとしても、このことを知らないため利用することができず、結局元のリソースサーバからダウンロードすることになる。一般に近傍のキャッシュリレー間は太い回線で接続されていることが多いため、お互いのキャッシュデータを利用しないのは非常に効率が悪いと考えられる。特にサイズの大きなファイルを遠方からダウンロードする場合などは、コスト的に非常に大きな無駄を払っていることになる。

4 問題点に対する解決のアプローチ

4.1 解決方針 1 ：適応型ストレージ管理

現行のキャッシュリレーシステムにおいては、制限サイズを越えたファイルは一切キャッシュされない。またキャッシュが一杯になりエントリーをどれか追い出さなければならない場合には、追い出しはファイル単位で行われてどれだけ大きいファイルであってもまるごと捨てられる。

これらの点を改良すべく、以下のような方針が考えられる。

- キャッシュリレーが limit-size を越えるファイルを受け取ったときには、ファイルの quality level を下げることによりファイルサイズを limit-size 以下に変更して、これをキャッシュする
- キャッシュの追い出しを行う際にも、大きなファイルはいきなりまるごと捨てず、まず quality level を下げることによりファイルサイズの変更を行う

4.2 品質レベルコントロール

動画や音声などのマルチメディアデータの場合には、quality level(QL) をコントロールすることが可能である。例えば動画であればピクチャフレームを適当に間引くようなことができる。また符合化を行う際の量子化ステップを大きくすることにより、全体の情報量を減らし、ファイルサイズを小さくすることもできる。さらに最近のマルチメディア符合化方式の中には、例えばMPEG2のようにスケーラビリティメカニズムを持つものもあり、そのような場合にはファイルの QL を容易に変更することできると考えられる。場合によってはファイルを単純に分割し部分的に保持するといったことも有効であろう。例えば動画データなどの場合、ファイルの先頭の部分を保持しておくことにより、少しだけ中身を覗いてみたいというユーザの要求に応えることができる。

4.3 品質レベルコントロールのキャッシュリレーへの適用

次に QL コントロールのキャッシュリレーへの適用を検討する。この際問題となるのは、QL を変更してキャッシュしたファイルに対し、次にクライアントからのリクエストが来た場合に、どのような対処をとるべきかということである。この場合キャッシュ中に存在するのは、リクエストされたオリジナルファイルと全く同じものではない。この時キャッシュリレーとクライアントがどのように対処すべきかということに関して、次のようなシナリオが考えられる。

1. クライアントはキャッシュリレーにリクエストを送る

2. キャッシュリレーはクライアントに、リクエストされたファイルを送り返す（またはリクエストされたファイルのQLをクライアントに通知する）
3. クライアントがそのファイルのQLに満足でない場合、キャッシュリレーに対して再リクエストを行い、キャッシュリレーはオリジナルファイルをリソースサーバから再ダウンロードする

QLの判断は、ユーザからの指示により決めることができる。ユーザが要求するQLをあらかじめ記録しておきこれに基づいて自動的に判断することも考えられるし、送られてきたデータをユーザが利用してみてこれに不満であったら再リクエストを行うという対話的操作も実現可能である。またクライアントのデバイス能力によって自動的に判断することも考えられる。例えば解像度の低いディスプレイを使っていた場合、その解像度を越える精密な画像データは必要なく、キャッシュされているファイルのQLがこれを満たすレベルであれば十分である。

キャッシュリレーにおけるQLのコントロールという本提案は、以下のような点で妥当なアプローチであると考えられる。

まず本提案において、QLコントロールはキャッシュリレー上だけで実現されており、リソースサーバ側は関与していない。すなわちリソースサーバに余計な処理を負わせる必要がない。このことは、負荷分散という観点からも、リソースサーバ側を現行のものから変更せずに実現できるという点からも望ましいことと考えられる。

また、キャッシュリレーにおけるリクエストファイルの仲介自体はリアルタイム性の高い仕事であるが、キャッシュのストレージ管理は急いで実行する必要はなく、CPUが空いた時にゆっくりと処理をすればよい。従ってこのメカニズムを加えたことにより、キャッシュリレーの仲介処理のパフォーマンスが低下することを心配する必要はない。

4.4 解決方針2：キャッシュリレーによるクラスタの形成

現行システムの問題点として二番目にあげたキャッシュリレー間の連携については、以下のような方針で対処できる。まず図2のように近傍のキャッシュリレーでクラスタを構成する。クラスタ内のキャッシュリレー間は十分に太いリンクで接続されている。そして個々のキャッシュリレーはそれぞれいくつかのクライアントを受け持っている。また図には示されていないが、あるクライアントから、そのクライアントの担当以外のキャッシュリレーに対して直接コネクションが張られていることもあり得る。

このクラスタ内で情報交換を行い、協調してキャッシュ動作を行うわけであるが、すべてのファイルに関して情報交換を行うのはオーバーヘッドが大きくなり好ましくない。そこであるスレーショールドレベルを設ける。このスレーショールドはファイルサイズやダウンロード時間などによって決めることができる。スレーショールド以下のファイルに関しては、通常のキャッシュリレー動作により処理される。

一方スレーショールドを越えるファイルをキャッシュした場合、クラスタ内の他のキャッシュリレーに対してハイパーリンク情報をマルチキャストする。ハイパーリンク情報は、ファイルがそのキャッシュリレーによりキャッシュされていることを示すアドレス情報である。これを受け取った他のキャッシュリレーは、自分のクライアントからそのファイルへのリクエストを受けた場合に、リクエストをハイパーリンク情報で示されたキャッシュリレーへと転送することができる。ファイルそのものをマルチキャストすることは、特にスレーショールドを越えるようなファイルの場合には、コストの面で好ましくない。しかしハイパーリンク情報は非常に小さい情報であり、これをマルチキャストしても、特に太いリンクで接続されているクラスタ内においてはほとんど問題にならないであろうと予想される。またマルチキャストする対象をスレーショールドを越えるファイルに関してのみと限定することにより、キャッシュリレーの負荷を最小限に押えながら、最も効果のあるキャッシュリレー間の協調動作を実現できる。

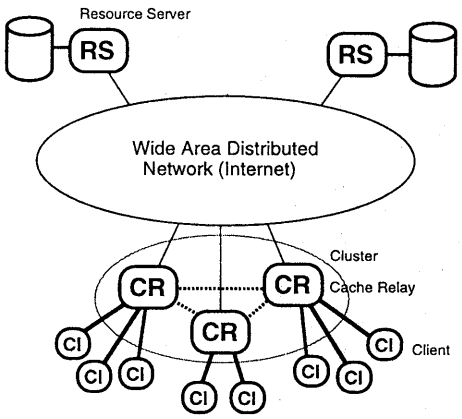

図 2: Formation of a cluster with cache relays

5 方針1、2に基づくキャッシュリレーメカニズムの実現

5.1 リクエストを受けた時のキャッシュリレーの動作

前節で検討した方針1、2を元に、本節ではキャッシュリレーメカニズムの実現方式の提案を行う。まずキャッシュにリクエストされたファイルが存在しなかった場合、すなわちそのファイルへの最初のアクセス時やキャッシュミスが起きた場合のキャッシュリレーの動作は以下のようになる。

1. クライアント (Cl) がキャッシュリレー (CR) 経由でリソースサーバ (RS) に対してリクエストを出す

2. RS から CR にファイルがダウンロードされ、Cl に渡される

3. ファイルが threshold レベル以下であった場合（このようなファイルを normal ファイルと呼ぶ）、ファイルはそのままキャッシュされ、threshold レベルを越えている場合にはクラスタ内の他の CR にハイパーリンク情報がマルチキャストされる（shared ファイルと呼ぶ）

4. ファイルが limit-size 以下であった場合、ファイルはそのままキャッシュされ、limit-size を越えている場合には QL コントロールによりファイルサイズを limit-size まで下げてからキャッシュする（reduced ファイルと呼ぶ）

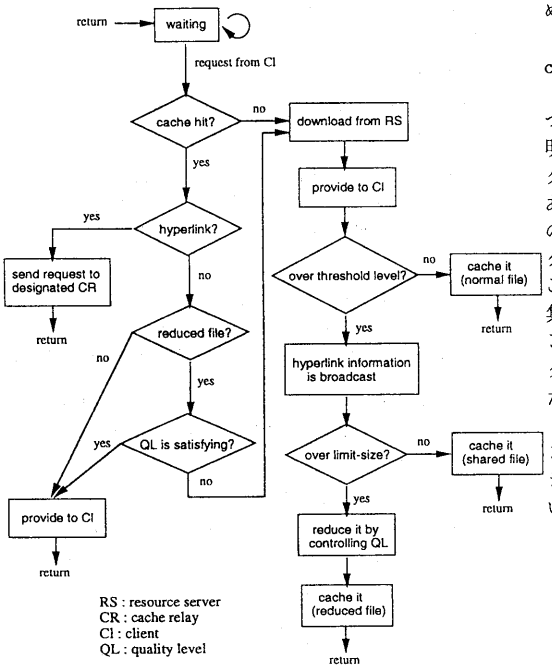

図 3: Flowchart of cache relay mechanism

一方、キャッシュにリクエストされたファイルが存在し、キャッシュヒットが起きた場合のキャッシュリレーの動作は以下のようになる。

1. CR に存在したのがハイパーリンク情報であった場合、リクエストは指定された CR に転送され、そこで再びキャッシュヒットの処理が行われる

2. CR にキャッシュされたファイルが normal ファイルであった場合、そのまま Cl に提供される

3. CR にキャッシュされたファイルが reduced ファイルであった場合、そのファイルの QL が満足のいくものであればそのまま Cl に提供される

4. もし QL が満足のいくものでなかったら、RS からオリジナルファイルを再度ダウンロードして Cl に提供し、その後上記のキャッシュミスの処理を行う

以上をまとめ、リクエストを受けた際のキャッシュリレーの動作をフローチャートで表すと図 3のようになる。

5.2 キャッシュエントリーのマネジメント

クライアントからのリクエストを受けた際の処理以外に、キャッシュリレーにおいてはエントリーのマネジメントも行う必要がある。これにはキャッシュが一杯になった時のエントリーの追い出し処理と、キャッシュしたデータとサーバに存在するオリジナルとの間で整合性を保つためのコンシステンシ処理とがある。

エントリーの追い出し処理には、一般に LRU(Least Recentry Used) 方式が用いられる。

また分散ファイルシステムにおいてコンシステンシを保つための手法としては、サーバ側からキャッシュに対して明示的に変更を通知する callback、キャッシュデータにアクセスがなされるたびにサーバに対してデータが最新であるかどうかを確かめる check-on-use、キャッシュデータの有効期限を定め、この期間が経過したらキャッシュデータは無効であるとする timeout などが知られている [6]。このうち WWW のような広域ネットワーク上での情報収集システムにおいては、サーバ側がキャッシュのマネジメントに関与するのはほぼ不可能であることや、ネットワークのトラヒックやアクセス時間ををできるだけ低く押えたいことから、一般に timeout 方式が用いられる。

従ってキャッシュリレーにおけるエントリーのマネジメントは基本的に LRU + timeout で処理される。まずキャッシュが一杯になった際の、LRU に基づくエントリーの追い出し処理の手順を以下に示す。

1. LRU(Least Recentry Used) であるファイルを特定する

2. ファイルが normal ファイルであった場合には、そのまま廃棄する

3. ファイルが shared ファイルや reduced ファイルであった場合には、さらに QL を下げることによりファイルサイズの縮小をはかる

4. QL が minimum レベルに達した場合には、ファイルを廃棄し、ハイパーリンク情報の消去命令をマルチキャストする

次にタイムアウトがきたファイルの処理の手順を示す。

1. ファイルが normal ファイルであった場合には、そのまま廃棄する

2. ファイルが shared ファイルや reduced ファイルであった場合には、ファイルを廃棄し、ハイパーリンク情報の消去命令をマルチキャストする

6 関連研究

文献 [7, 8] では静止画像の階層的符合化 [9] を利用した分散型階層的画像データベースシステムの実現方式が提案されている。これは画像データベースシステムのセンターサーバからサブノードへキャッシュしたデータを、時間の経過と共に少しずつ削除していくシステムの提案である。このシステムにおいては静止画像の階層的符合化が用いられているため、一部が削除された少ないデータ量でも分解能の荒い画像を得られることができるようになっている。この実現方法においては、本稿で解決方針1として述べた適応型ストレージ管理のアイディアと基本的な部分で同じ点に着目がなされている。我々はこのようなアイディアは、データベースシステムのように固定的なサーバとクライアントから構成される閉じた世界より、広域分散環境における情報検索・収集システムにおいて、より有効であると考えている。

また文献 [10] においては、WWW におけるキャッシュのアルゴリズムとして、過去のアクセスパターンに基づいてキャッシュのマネジメントを行う手法が提案されている。この文献においては、人の記憶メカニズムの心理学的研究において用いられるモデルを利用し、実際の WWW サーバへのアクセスログからユーザのアクセス傾向を解析して、どのようなキャッシュマネジメントを行えばヒット率を高くすることができるのかといった検討がなされている。このようなデータのプリフェッチやマネジメントの工夫によりキャッシュのヒット率を高めるためのアイディアは、今後大いに研究されていくと思われる事柄であり、本稿で提案したキャッシュリレーメカニズムにもそれらのアイディアが組み合わされていくべきであろうと考える。

7 おわりに

本稿では広域分散ネットワーク上での情報収集システムにおけるキャッシュリレーメカニズムの検討を行った。まず最初にキャッシュリレーの概念を簡単に説明した後、現行のキャッシュリレーの実現方式においては、サイズの大きなファイルがキャッシュされないという点と、近傍のキャッシュリレー間で連携がなされていないという点が問題であることを指摘した。そしてその解決策として、品質レベルのコントロールによるファイルサイズの変更をキャッシュリレーで行うこと、および近傍のキャッシュリレー間でクラスタを形成し、ハイパーリンク情報を用いて連携を行うことを提案した。そしてこれらの方針を組み合わせたキャッシュリレーメカニズムの実現方式を示した。

キャッシュリレーメカニズムの実現方式にはさまざまなバリエーションが考えられる。例えば発展方式の一つとして、サーバ側にも品質レベルコントロールの機能を持たせることが考えられる。すなわちキャッシュリレーからサーバにリクエストをする際に品質レベルも一緒に指示し、この品質レベルコントロール処理をサーバで行いあらかじめサイズを小さくしたファイルをキャッシュリレーにダウンロードするような仕組みである。本稿で提案したのはキャッシュリレーだけで品質レベルコントロールの処理を行う方式であるため、サーバ側は現行のものを改変する必要がなく、このことは World-Wide Web のようなオープンで分散した情報収集システムにおいて非常に大きなメリットであると考えられる。しかし上述のようにサーバにも品質レベルコントロール機能を付け加えサーバにおいてもファイルサイズを変更することができるようにすれば、サーバからキャッシュリレーへとダウンロードするデータ量を減らすことができ、アクセス時間とトラヒックの両面から好ましいと考えられる。今後は提案方式の評価や試験的な実装などをすると共に、モデルをさらに発展させるための検討を行う予定である。

参考文献

[1] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Nielsen and A. Secret, "The World-Wide Web", *Communications of the ACM*, Vol.37, No.8, pp.76-82, August 1994.

[2] A. Luotonen and K. Altis, "World-Wide Web Proxies", *First International World Wide Web Conference*, May 1994.

[3] E. D. Katz, M. Butler, and R. McGrath, "A Scalable HTTP Server: The NCSA Prototype", *Computer Networks and ISDN Systems*, Vol.27, pp.155-164, 1994.

[4] T. T. Kwan and R. E. McGrath, "NCSA's World Wide Web Server: Design and Performance", *IEEE Computer*, Vol.28, No.11, pp.68-74, November 1995.

[5] S. Glassman, "A Caching Relay for the World Wide Web", *First International World Wide Web Conference*, May 1994.

[6] B. C. Neuman, "Scale in Distributed Systems",*Readings in Distributed Computing Systems*, pp.463-489, IEEE Computer Society Press, 1994.

[7] 片山文雄、安野貴之、安田靖彦、"人の記憶モデルに基づく分散型階層的画像データベースの一方式"、画像電子学会1995年次大会予稿、pp.20-21, June 1995.

[8] 片山文雄、樋田隆、安野貴之、安田靖彦、"人の記憶モデルに基づく分散型階層的画像データベースの実験的検討"、1995年画像符合化シンポジウム、pp.21-22, October 1995.

[9] 安田靖彦、高木幹雄、加藤茂夫、粟野友文、"階層的符合化法による静止画像の段階的伝送および表示"、信学論 (B), Vol.J63-B, No.4, pp.379-386, April 1980.

[10] J. E. Pitkow and M. M. Recker, "A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns", *Second International World Wide Web Conference*, October 1994.