フィンガープリントキャッシュを用いた動的Webコンテンツの効率的な配信方式

庄野篤司[†] 佐藤英昭[†] 木村康浩[†]
木場雄一[†] 関俊文[†]

Webを快適に利用するためにキャッシュは欠かせない技術である．しかしWebの利用分野が広がるとともに，従来のキャッシュでは扱えない動的コンテンツが増加している．動的コンテンツはユーザからのリクエストごとに新しく生成したり認証したりして選択するので，同一のコンテンツや類似のコンテンツが多いという性質を持つ．本論文ではこの性質を利用してネットワーク上を転送するデータ量を削減する方式について述べる．コンテンツの同一性を判断するために，コンテンツのダイジェスト値であるフィンガープリントをキーにしてコンテンツを管理するフィンガープリントキャッシュを用いる．転送するコンテンツからフィンガープリントを計算し，それと同じコンテンツがフィンガープリントキャッシュにあれば，フィンガープリントだけを転送する．類似のコンテンツの場合は，フィンガープリントキャッシュに管理されているコンテンツを辞書として利用し，転送するコンテンツを差分圧縮する．本手法によるデータ量削減効果と低速回線時のレスポンスタイム向上効果の評価を実施した．

Efficient Dynamic Web Contents Delivery Method Utilizing Fingerprint Cache

Atsushi Shono,[†] Hideaki Sato,[†] Yasuhiro Kimura,[†]
Yuuichi Koba[†] and Toshibumi Seki[†]

Caching is an indispensable technique for reducing Web traffic. However, widely used proxy cache or browser cache is helpless against recently increasing dynamic Web contents. This article proposes a novel method to reduce network traffic with dual proxy system. We utilize a new caching scheme named fingerprint cache. Fingerprint is a digest value calculated from a content. Every dynamic Web contents are stored in the fingerprint cache with their fingerprints. To suppress sending same contents again, server side proxy replaces them with their fingerprints. Client side proxy restore them conferring its fingerprint cache. To compress contents which resemble to previously transferred contents, server side proxy uses differencial compression technique. Contents in the fingerprint cache are utilized as a dictionary for compression and decompression. Web show availabilities of our own method for compressing data and reducing TTD (Time to Display) in a low bandwidth network.

1. はじめに

急速に増加し続けるインターネット上のWebトラフィックを軽減する手段としてWebキャッシュは欠かせない技術である[1)]．しかし従来のWebキャッシュは更新頻度の低い静的コンテンツのみをキャッシュし，Webの応用領域の広がりとともに増加している動的コンテンツはキャッシュできない．そこで本論文では，コンテンツの内容をキーにしてキャッシュするフィンガープリントキャッシュと呼ぶ機構を利用して，動的コンテンツを効率良く配信する方式を提案する．

Webキャッシュは過去にアクセスしたコンテンツを保存しておき，次に同じコンテンツのリクエストがあれば保存しておいたコンテンツを返すように動作する．その結果，クライアントから見た応答時間を短縮するとともに，ネットワーク上のトラフィックを抑え，さらにサーバの負荷を低減する効果が得られる．従来からのWebの利用方法では，Webサーバは登録されている静的コンテンツをリクエストに応じて単純に送り返す場合が大半を占めている．そのため，それらの負荷をキャッシュで軽減することで大きな効果を得ることができた．しかし，ポータルサイトやEC（Electronic Commerce）サイト，さらにはWebをフロントエン

† 株式会社東芝研究開発センター
Toshiba Corporate Research & Development Center

ドとした企業内業務プログラムなど，Web の利用形態が広がるにつれて静的コンテンツだけでは構築できない Web サイトが増加している．

動的コンテンツはクライアントからのリクエストを受け取るたびにコンテンツを生成あるいは選択して送り返されるため，原理的にキャッシュすることはできない．しかし，実際にはリクエストごとにまったく異なるコンテンツを送り返すことは少なく，同一コンテンツや類似コンテンツを送り返すことが多い．

そこで，過去に送ったコンテンツと新しく取得したコンテンツの差分のみを送ることで HTTP レスポンスのサイズを削減する研究が行われている．HTTP で取得するコンテンツの差分（delta）を送ることで HTTP レスポンスを削減するというアイデアは文献 2) で初めて報告された．文献 2) は HTTP[3),4)] の POST メソッドに対する各レスポンス（動的コンテンツ）が非常に類似している点に着目し，変化した部分（差分）のみを送ることで転送データ量を削減した．

文献 5) では delta-encoding が動的コンテンツ配信において非常に有効な HTTP レスポンスサイズ削減手段であることを定量的に評価，その潜在的な有効性を確認をした．また，文献 6)，7)，8)，9) では delta-encoding を利用した様々な動的コンテンツのサイズ削減についての研究が行われた．

また，文献 10) および 11) では Cache Digest と呼ばれるプロトコルが提案されている．これは squid のように親子・兄弟の存在するプロキシ間で ICP などによって互いがキャッシュしているコンテンツを通知・交換するためのものである．しかし，これは静的なコンテンツを対象にしており，Cache Digest を用いて互いに交換するのは URL のダイジェスト値である．動的コンテンツを対象とする提案方式では URL をコンテンツの識別子として用いないため，単純に Cache Digest を適用することはできない．

本論文ではフィンガープリントキャッシュを用いた 2 種類のデータ圧縮手法を示し，そのデータ圧縮効果と低速回線の場合の表示時間向上効果の評価を示す．2 章では動的コンテンツについて述べ，3 章では従来の動的コンテンツの転送データ量削減技術について示す．4 章では我々が提案する動的コンテンツ削減方法を示す．5 章で本論文の方式の性能評価を行い，6 章で提案する手法の適用先と拡張方法について述べる．最後に 7 章で結論を述べる．

2. 動的コンテンツ

2.1 従来キャッシュに保存できないコンテンツ

Web において従来から広く使われているキャッシュには，Web ブラウザが持つブラウザキャッシュと，プロキシサーバが持つプロキシキャッシュがある．HTTP のプロトコル上，これらのキャッシュはどのようなコンテンツでもキャッシュできるわけではない．キャッシュできないコンテンツは，大きく次の 2 つに分類できる．

(1) **Web サーバが no-cache 指定したコンテンツ**

Web の転送プロトコルである HTTP は，コンテンツがキャッシュ可能かどうかを指定するリプライヘッダを用意している．Web サーバはキャッシュしてほしくないコンテンツを返す際には，

```
Cache-Control: no-cache
```

という形式のキャッシュ禁止指示ヘッダ☆を付ける．従来のプロキシキャッシュやブラウザキャッシュはこのヘッダを持つコンテンツをキャッシュしてはいけない．

(2) **アクセス認証されたコンテンツ**

HTTP ではユーザを認証するために，

```
Authorization: 認証情報
```

という形式の認証ヘッダを用意している．認証ヘッダを付けて送ったリクエストのリプライは認証されたユーザ以外はアクセスしてはいけないので，複数のユーザで共有するプロキシキャッシュはキャッシュしてはいけない．ただし，ブラウザキャッシュは他のユーザと共有しないのでキャッシュしてもかまわない．

2.2 動的コンテンツ

本論文では，従来のプロキシキャッシュでキャッシュできるコンテンツを静的コンテンツ，反対にキャッシュできないコンテンツを総称して動的コンテンツと呼ぶ．

Web の利用が広がるにつれて，動的コンテンツの利用が増加している．代表的な使い方には以下のような例がある．

- **ポータルサイト：**ユーザの好みなどのプロファイル情報を管理し，その情報を利用して個々のユーザ向けにカスタマイズしたページを提供する．
- **電子商取引：**セキュリティを保ちつつ，取引に必要な情報を会話的にやりとりする．
- **リアルタイム情報サービス：**最新の株価やニュースなど頻繁に更新される情報を提供する．
- **有料コンテンツ：**音楽やソフトウェアなどのデジタルコンテンツを電子的に流通させる．

☆ 古いバージョンの HTTP では `Pragma` で指定する．

- **企業内システム：**企業内の業務プログラムを Web ブラウザから使えるようにする．

これらのシステムの多くは次のように実装されているため，従来のプロキシキャッシュではキャッシュできない動的コンテンツを多く使用する．

(1) ユーザからリクエストが来ると CGI や JAVA のサーブレットなどの機構を使ってアプリケーションプログラムを起動する．起動されたプログラムは，必要に応じてデータベースから検索したデータを使ってコンテンツを生成し，それをユーザに送り返す．

(2) パスワードなどを用いてユーザを認証し，そのユーザのアクセス権に応じてアクセスを制限したり，あるいはユーザごとに異なるコンテンツを選択して送り返す．

2.3 動的コンテンツの特徴

本論文で扱う動的コンテンツには以下にあげる大きな 2 つの特徴がある．

- **URL は異なるのに同じコンテンツ：**このケースは HTML の広告用のインラインイメージなどがよい例である．この特徴は aliasing と呼ばれている[12]．
- **同じ URL なのに異なるコンテンツ：**このケースは 2.2 節で示したように CGI やサーブレットなどといったページ生成プログラムを用いて生成されたコンテンツが該当する．この特徴は resource modification と呼ばれており，delta-encoding が有効に働く[2),5),9)]．

aliasing は Web サーバの構成の改善により減少させることが可能である．たとえばインラインイメージを様々な URL にリンクしていたものを Web サーバ側の一 URL に集約することで静的なキャッシュを有効に活用することがあげられる[13]．しかし，既存の Web サイトの変更を余儀なくされ，この回避方法は新しく構築される Web サイトにしか有効ではない．aliasing による HTTP データの冗長を避ける必要がある．

また，resource modification は Web の普及にともない増加の一途をたどっている．resource modification が発生した場合，多くの場合は各レスポンスに共通した静的な部分と各レスポンスごとに異なる動的な部分から構成されている．この静的な部分がレスポンスのたびに毎回ネットワークを流れる冗長を避ける必要がある．

2.4 高速化技術の必要性

以上述べたように動的コンテンツには静的なキャッシュが効かず，HTTP データの冗長によるネットワーク遅延が発生する．エンドユーザにはネットワーク遅延による待ち時間の増加や，無線ネットワークをはじめとする従量課金の環境下ではコスト増といった問題が発生する．

しかし，近年のブロードバンド技術の普及は著しい．xDSL や光回線を用いた常時接続サービスや携帯電話網による定額サービスが一般的になり，HTTP データの冗長によるネットワーク遅延やコスト増加によるエンドユーザへの負担は大幅に改善されてきた．また，企業においてもブロードバンドを活用し，SSL（Secure Socket Layer）や VPN（Virtual Private Network）を用いてセキュアにリモートアクセスを実現できるようになってきた．

それにもかかわらず，本論文で示すような動的コンテンツの高速配信技術に対するニーズは高い．たとえば企業内での Web をフロントエンドとした定型業務などがあげられる．ブロードバンドを利用したリモートアクセスは，SSL や VPN 技術といったセキュリティ技術を適用しているとはいえ，顧客情報や機密情報といった重要情報を公衆回線に流すリスクがある．そのため，専用線を用いている場合が多い．また，ブロードバンドを利用せずに従来のモデム接続を行っているエンドユーザ数も多い[9]．さらに，近年 P2P [14] によるインターネット上の通信量の増加が著しく，インターネット帯域の多くが P2P によって使用されている．そのため，Web 閲覧の際にコンテンツの取得に時間がかかるなどの問題が発生している．本論文で述べる HTTP の通信データ量の削減による高速化技術はそのような問題に対しても有効に機能すると考えられる．

3. 従来の動的コンテンツ配信高速化方式

動的コンテンツの増加によって，従来のキャッシュ以外の Web システムの性能改善手段が必要になってきている．原理的には，動的コンテンツは必ず Web サーバにリクエストを送って処理した結果を受け取らなければならない．そのためサーバの負荷が問題になる場合には，より高性能な計算機への置き換えや負荷分散装置と計算機クラスタの組合せなどで，Web サーバの処理能力を向上させることが多い[15]．しかしネットワークのトラフィックの増加が新たなボトルネックになるため，以下に説明するようにネットワークの負荷を軽減するいくつかの方式が提案されている．

3.1 データ圧縮

動的コンテンツにも静的コンテンツにも適用できる手法としてデータ圧縮がある．代表的な手法としては，転送データを汎用の圧縮形式（gzip など）を使って圧

縮する方法，画像データをプロキシサーバで変換して圧縮率を上げる方法[16),17)] がある．

3.2 動的コンテンツのキャッシング

通常はキャッシュ不可能な動的コンテンツをキャッシュすることで，サーバやネットワークの負荷を軽減する方式はいくつか提案されている．

データの一貫性を崩さない方式としては，動的コンテンツを生成したアプリケーションがキャッシュの無効化を指示する方式がある[18)]．たとえばデータベースを参照して動的コンテンツを生成した場合，参照したデータベースが更新されたことをデータベース管理システムが検出すると，キャッシュに対して保存しているコンテンツの無効化を指示する．この方式は，キャッシュの無効化を指示する機能をアプリケーションプログラムやデータベース管理システムに埋め込む必要があるほか，無効化を指示するためのトラフィックが生じるという欠点がある．

一貫性を弱めた方式としては，たとえば動的コンテンツに生存時間（time to live）を付加して，一定期間のキャッシングを許す方式[19)] もある．しかしこの方式は，データ更新がすぐに反映されなければならないような強い一貫性を要求するアプリケーションには利用できず，用途が限定される．

また，MD5[20)] や SHA[21)] などのハッシュ関数を用いた方法も考案されている[7),9)]．これらの研究の基本的なアイデアは，従来の静的なキャッシュが URL をキーにしてコンテンツをキャッシュしているのに対し，コンテンツのダイジェスト値をキーにしてコンテンツをキャッシュするというものである．ダイジェスト値をキーにすることで，リクエストのたびに生成される動的コンテンツを一意に識別できる．文献 7) ではコンテンツ全体のダイジェスト値を利用するのに対し，文献 9) ではコンテンツを約 2 KB のブロックに分割してブロック単位でダイジェスト値を用いている．このアイデアを用いることで，aliasing による冗長を回避することができる．4 章で示すように我々の方式でもこのアイデアを利用している．

3.3 差分転送

過去に転送した類似のコンテンツとの差分（delta）を転送して受け取り側で復元できれば，転送データ量を減らすことができる．差分転送が有効に働くためには，送りたいコンテンツとできるだけ似ているものを差分をとるベースコンテンツとして選ぶ必要がある．最も簡単で広く使われている方式[2),5)] は，過去に同じ URL のリクエストが返したものをベースコンテンツにする方式である．しかし次のような場合には単純に URL だけでは類似性を判断できない．

(1) 同じ URL でもユーザごとに異なるコンテンツを返す場合，同じユーザの過去のコンテンツとは類似性が高いが，他のユーザのものとは類似性が低い．個人の嗜好に合わせてパーソナライズする Web サイトではこのようなコンテンツが多い．

(2) URL は異なるが同じレイアウトで情報を記述している場合，それらの URL 間では類似性が高い．

そこで，類似する URL のグループを設定し，そのグループで共通に用いられるベースコンテンツをグループに属するコンテンツの共通部分から合成する手法が提案されている[8)]．しかし，Web サイト全体の構成を把握したうえで設定を行わなければならない．また，ベースコンテンツは各クライアントに配信されるため，ベースコンテンツに個人情報が含まれる可能性を排除しなければならない．文献 8) ではベースコンテンツのサイズ☆とのトレードオフでその可能性を排除しているが，ベースコンテンツの選定・合成に複雑なステップを踏む必要がある．

文献 7) では過去に取得したコンテンツをベースコンテンツとしており，5 章に示す我々の提案する方式と非常に類似している．しかし，delta-encoding として GDIFF[22)] を採用している点，ベースコンテンツが 1 つという点などで我々の方式と異なる．

文献 9) はコンテンツを約 2 KB のブロックに分割して管理する手法を採用している．もしすでにサーバ側プロキシが Web サーバからキャッシュ済みのブロックを含むコンテンツを取得した場合，該当ブロックはそのダイジェスト値のみをクライアント側に送信する．キャッシュしていないブロックはオリジナルデータとともにサーバ側プロキシで計算したダイジェスト値をクライアント側に送信する．文献 9) ではこのブロック単位の管理を実現するために Rabin 関数[23)] を用いた手法を導入した．それによりこの方式の最大の問題点であったブロックの境界の決定を効率良く行うことに成功し，上記ダイジェスト値の転送による HTTP レスポンスの削減を実現した．

4. FP キャッシュを用いた Web アクセス高速化技術

動的コンテンツはユーザからのリクエストごとに生成されたり，あるいはユーザを認証して選択される．そのため，まったく異なるコンテンツばかりが送り返されることは少なく，同一のコンテンツである場合や

☆ サイズが大きいほど delta-encoding の効果が大きくなる．

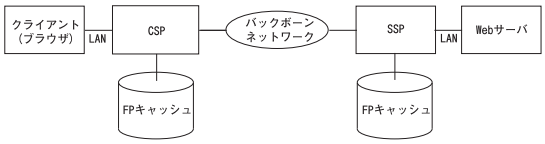

図 1 システム構成
Fig. 1 System configuration.

類似のコンテンツである場合が多い．そこで本論文で述べる手法は，この性質を生かして転送データ量を抑える．FP 転送（4.4 節）によって aliasing による無駄なトラフィックの発生を抑えるとともに，差分転送（4.5 節）によって resource modification によるトラフィックの増加を抑える．

4.1 システムの構成

本論文で提案する方式は，**図 1** に示すように，Web サーバとクライアント（Web ブラウザなど）の双方にサーバ側プロキシ（以後，SSP と呼ぶ）とクライアント側プロキシ（以後，CSP と呼ぶ）と呼ぶプロキシサーバを設ける．ただし，CSP は専用サーバとして実現する場合と，クライアント PC のアプリケーションによるローカルプロキシとして実現する場合がある．以後は簡略化のために前者を用いて説明を行う．Web サーバと SSP は高速な LAN（Local Area Network）で接続されており，専用サーバの場合はクライアントと CSP も高速な LAN で接続されているものとする．SSP と CSP の間はインターネットや WAN で接続されるので，一般に LAN ほどの高速性は期待できない．クライアントと Web サーバ間の通信にはこの 2 つのプロキシサーバが介入し，4.4 節で説明する同一のコンテンツの圧縮と 4.5 節で説明する類似コンテンツの圧縮を行うことでネットワークのトラフィックを抑える．

4.2 FP キャッシュ

SSP と CSP の間で過去に転送したものと同一のコンテンツを再度送らないようにするためには，コンテンツの同一性を判断する手段が必要になる．動的コンテンツの性質上，URL が同じであっても同一のコンテンツであるとは限らない（resource modification）．また，異なる URL であっても同一のコンテンツであることもある（aliasing）．そこでコンテンツの内容から一意に計算して得られるダイジェスト値をキーにしてコンテンツを記録するキャッシュを利用する．ダイジェスト値の計算法には様々な方式が考えられるが，同一コンテンツなら同じ値を与え，かつ異なるコンテンツなら高い確率で異なる値を与えるものが望ましい．そのような性質を満たす関数としては MD5 [20] や SHA1 [21] などのハッシュ関数を用いることができるが[7),9)]，本論文で示す手法では MD5 を採用した．

本論文ではコンテンツそのもののダイジェスト値と付加的な値から構成される値をフィンガープリント（以後，FP と呼ぶ）と呼ぶ．FP は 64 バイトの文字列であり，コンテンツのダイジェスト値から構成されることからコンテンツを一意に識別する．また，FP をキーにしてコンテンツを管理するキャッシュをフィンガープリントキャッシュ（以後，FP キャッシュと呼ぶ）と呼ぶ．

また，FP キャッシュは認証を経たコンテンツもキャッシュ対象とする．FP キャッシュを搭載する SSP と CSP に不正にアクセスされることで機密情報の漏洩が考えられる．SSP は ISP や Web サーバ側 LAN 内に設置されるため，適切なアクセスコントロールを行うことでそのセキュリティを確保することができる．また，CSP は専用サーバとして実現する場合は SSP と同様に十分なセキュリティ対策を講じればよい．ローカルプロキシとして実現する場合は，キャッシュを暗号化することで情報漏洩の危険性を回避できる．

4.3 静的キャッシュとしての動作

SSP や CSP は，FP キャッシュを利用して，静的コンテンツをキャッシュする従来のプロキシキャッシュとしても動作する．その場合，FP キャッシュに加え過去にアクセスした各コンテンツごとにその FP と MIME タイプ[24)] やコンテンツの新鮮さ，最後にアクセスした時間とキャッシュ可能な HTTP ヘッダを記載したテーブルを併用する．このテーブルはサーバから送られてきたコンテンツに no-cache 指定がなく，かつ，それに対応するリクエストに認証ヘッダがない場合に生成される．各テーブルは該当する URL の文字列から MD5 で算出したダイジェスト値によって名前付けされ URL ごとに管理される．

SSP と CSP は，クライアントから新しくリクエストを受けると，その URL のダイジェスト値を計算し，該当するテーブルがあるかを検索する．テーブルがあった場合は通常のプロキシキャッシュと同様にコンテンツの新しさを調べ，十分に新しいと判断すれば FP キャッシュからコンテンツを，上記テーブルからヘッダを取得して送り返す．なかった場合は上流にリクエストを転送し，レスポンスを受信したら 4.4，4.5 節に示すデータ削減を行う．

4.4 FP 転 送

本節では SSP と CSP が連携し，FP キャッシュを利用して同一コンテンツの転送トラフィックを抑える処理（以後，FP 転送と呼ぶ）について示す．CSP は

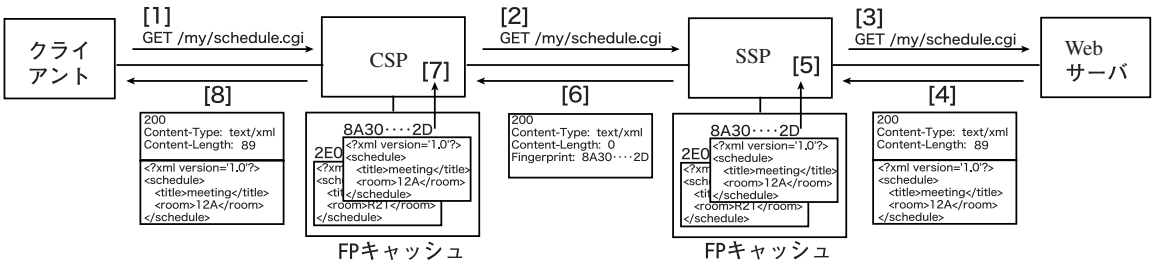

図 2 同一コンテンツの FP への置換
Fig. 2 Replacement of same contents with fingerprints.

上流にリクエストを転送する際に X-FP-Client ヘッダを追加する．SSP はリクエストヘッダに上記ヘッダがある場合のみ本節および次節に示す圧縮転送を行う．FP 転送の手順は以下のようになる．

(1) クライアントが出した HTTP のリクエストは，CSP と SSP を経由して Web サーバに伝えられる（**図 2** の [1][2][3]）．

(2) Web サーバでは，必要に応じてユーザの認証やアクセス権のチェックを行い，CGI やサーブレットなどのプログラムを実行して，結果のコンテンツを SSP に送り返す（図 2 の [4]）．

(3) SSP はサーバから受け取ったコンテンツの FP を計算して，同じコンテンツが FP キャッシュにあるかどうか調べる（図 2 の [5]）．

(4) FP キャッシュにあれば，HTTP のヘッダにその FP を指定する独自の Fingerprint ヘッダを追加し，HTTP ボディは空にして HTTP レスポンスとして CSP に送り返す（図 2 の [6]）．これによりコンテンツそのものではなく 64 バイトの FP を転送することによって転送データ量を削減する．

(5) CSP は指定された FP のコンテンツを FP キャッシュから取得し，Fingerprint ヘッダを削除したうえでクライアントに送り返す（図 2 の [7][8]）．

なお，(4) で FP キャッシュになかった場合は，SSP はそのコンテンツをそのままあるいは 4.5 節で述べる差分圧縮した形式で CSP に送り返す．それと同時に，SSP と CSP の双方の FP キャッシュに，計算した FP をキーにしてそのコンテンツを登録する．

FP はハッシュ関数を用いて計算するため，非常に小さな確率ながら，異なるコンテンツに対して同じ値を与える可能性がある．このような誤った判断を回避するには，上記のステップ (4) で FP キャッシュにあった場合でも，それを読み出して本当にサーバから受信したコンテンツと同じかどうかを比較して判断すればよい．

なお，HTTP/1.1[4] では Etag ヘッダと If-None-Match ヘッダを用いて動的コンテンツによるトラフィックの増加を抑制するメカニズムが採り入れられている．これらのヘッダによって resource modification や aliasing を検知し，なにかしらのデータ削減手法を行うことが可能であると考えられる．しかし，Etag は Web サーバが付加する値であり，後方互換を考慮し我々は独自に FP というコンテンツ識別子を導入した．

4.5 差分転送

動的コンテンツには，データベースから検索したデータを共通のレイアウトに埋め込んだコンテンツのように，同一ではないが類似性の高いものも多い．また，異なる URL からの類似コンテンツもある．本節ではそのような類似コンテンツを FP キャッシュを利用して圧縮して転送する方式（差分転送）について述べる．

まず，本節で示す差分圧縮の対象となるコンテンツは HTML やスタイルシートなどといった MIME タイプが text/*のテキストコンテンツに限定する．ページを構成するコンテンツのうち装飾などに用いられる画像は多くの場合は静的なコンテンツであり，4.3 節に示した静的なキャッシュとしての機能やブラウザのキャッシュによって冗長な転送は回避することができる．また，広告用の画像などは aliasing を起こす場合が多く，4.4 節に示した FP 転送でその冗長な転送を削減することができる．

類似コンテンツの差分圧縮を有効に働かせるためには，送りたいコンテンツとできるだけ似ているベースコンテンツを選ぶ必要がある．従来の方式ではベースコンテンツは過去に転送したコンテンツそのものであったり[2]，URL の分類に基づいて合成されたものであったりした[8]．この場合，差分の算出方法として diff や vdelta[25] などを用いていた[26]．それに対し，我々は行単位のキャッシュを用いた行単位の差分を実現した．**図 3** にその概念図を示す．

基本的な考え方としては，コンテンツの圧縮に用いる辞書として FP キャッシュに登録されているコンテ

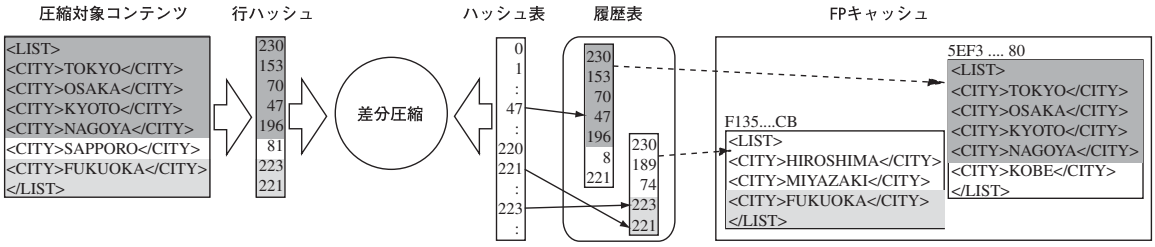

図 3 差分圧縮の概念図
Fig. 3 Conceptual figure of delta-encoding.

FP: 5EF3...80 Offset: 0 Length: 84	Length: 21 <CITY>SAPPORO</CITY>	FP: F135...CB Offset: 52 Length: 29

図 4 差分圧縮後のデータ構造
Fig. 4 Data structure of delta.

ンツを使う．圧縮対象コンテンツから FP キャッシュに登録されているコンテンツと同じ部分を検出し，その部分を FP キャッシュにあるコンテンツの FP とオフセットおよび長さで置き換える．また，一致する部分が見つからなかった部分はそのまま残す．図 3 の圧縮対象コンテンツは本節で示す差分圧縮処理により後述の**図 4** のフォーマットをした圧縮データに変換される．以下に図 3 を参照しながら差分圧縮の処理を示す．

図 3 は過去に FP がそれぞれ 5EF...80，F135...CB の 2 つのコンテンツの差分圧縮を行った結果，FP キャッシュと後述の履歴表およびハッシュ表にそれぞれの値が登録されている状態である．メモリ上にある履歴表にはそれぞれのコンテンツを行単位に分割し，各行を後述のハッシュ関数によりハッシュ値に置き換えた行ハッシュ値列が登録されている（図の履歴表）．圧縮対象コンテンツは差分圧縮処理にかけられる際にまず行ハッシュ値列を計算する（図の行ハッシュ）．次に算出された行ハッシュ値列と履歴表にある行ハッシュ値列を比較することですでに FP キャッシュに登録されているコンテンツと同一な部分を検索する．検索に際してハッシュ表を用いて高速に検索を行う（図のハッシュ表）．図の例では圧縮対象コンテンツの濃い網かけ部分と FP が 5EF3...80 のので示されるコンテンツの濃い網かけ部分が一致し，圧縮対象コンテンツの薄い網かけ部分と FP が F135...CB の薄い網かけ部分が一致する．また，圧縮対象コンテンツの網かけのない部分は一致するものが見つからなかったことを示す．一致する部分はその FP とオフセットおよび長さで置き換えることが可能であり，最終的に図 4 に示すフォーマットの差分圧縮データに変換される．この差分圧縮データの先頭部分は FP 値 5EF3...80 をキーにしてキャッシュされているコンテンツの 0 バイト目から 84 バイト分をコピーすることを示し，その後に 21 バイトの文字列をそのまま保持し，最後は FP 値 F135...CB をキーにしてキャッシュされているコンテンツの 52 バイト目から 29 バイト分をコピーすることを示している．最後に圧縮対象コンテンツを FP キャッシュに，その行ハッシュ値列を履歴表に登録し，次回以降の差分圧縮に利用できるようにする．その際，ハッシュ表も更新する．次に，上記で説明した差分圧縮処理に採用した高速化・最適化手法について述べる．

(1) **同一部分の検索の高速化**

同一部分の検索を行う際に，単純に 1 文字ずつ比較するのは効率が悪い．また，FP キャッシュに登録されている全コンテンツを対象とすると膨大な数の比較処理が必要となり，実際的なパフォーマンスが得られない．そこで，コンテンツを行ごとに分割して各行のハッシュ値どうしを比較する方法を採用した．ここで用いるハッシュ関数は入力文字列に対して各文字の加算とビットシフトおよびビットマスクによって履歴表の総行数と同じ値を上限とするハッシュ値を算出する．履歴表はリングバッファになっており，たとえば 64 K 行のリングバッファならハッシュ関数は 0–65535 のハッシュ値を算出する．FP キャッシュに登録されているコンテンツの行ハッシュ値列と圧縮対象コンテンツの行ハッシュ値列を比較することで文字単位の比較に比べて比較回数を削減する．また，FP キャッシュに登録されているコンテンツの行ハッシュ値列をメモリ上にある履歴表に登録しておく．メモリ上に登録しておくことでディスク I/O をともなわない高速な行ハッシュ値列の検索が可能となる．さらに，履歴表をリングバッファとすることで比較対象を最近取得したまだ履歴表にあるコンテンツに限定し検索対象コンテンツ数を削減する．また，ハッシュ表を用いることで同一行の検索をさらに高速化する．ハッシュ表の値の範囲は上述のハッシュ関数と同じである．ハッシュ表

の更新により，同一の行を持つコンテンツが新たに履歴表に登録された場合，新しいコンテンツが選択されるようになる．これは動的なページはその前後のページとの類似性が高いという一般的にいわれている特徴を考えると効率的である．

(2) **ベースコンテンツの統合**

履歴表とハッシュ表を用いた検索により圧縮対象コンテンツの置き換え可能な部分の検出が終わっている．しかし，多くのコンテンツをベースコンテンツとしている可能性がある．そこで，もしベースコンテンツ間に重複している部分があれば 1 つのベースコンテンツで表現するようにベースコンテンツの統合処理を行う．この統合処理によって置き換えに利用するベースコンテンツの数をできるだけ少なくすることができる．

なお，行ハッシュ値列は異なる文字列に対して同じ値を与えてしまう可能性がある．そこで，置き換える部分は FP キャッシュにある実際のコンテンツと比較することで誤った差分データを生成することを避ける．もし比較の結果誤りがあった場合，その部分は置き換えを行わずにそのままの文字列（図 4 の ⟨CITY⟩SAPPORO⟨/CITY⟩ と同じ形式）として差分データに変換される．また，もし差分圧縮後の差分圧縮データのサイズが差分圧縮前のコンテンツサイズより大きくなってしまった場合は差分処理は失敗とし，SSP は FP キャッシュと履歴表およびハッシュ表への登録後に Fingerprint ヘッダにその FP を挿入してオリジナルのコンテンツを下流に転送する．

図 4 のように差分圧縮されたコンテンツは，復元後の FP を Fingerprint ヘッダに挿入し HTTP レスポンスのボディとして SSP から CSP に転送される．また，SSP と CSP の間にあるプロキシなどにキャッシュされることを避けるために Cache-Control: no-cache も挿入する．元の Cache-Control ヘッダなどは専用のヘッダに退避したうえで下流に転送する．差分圧縮されたコンテンツは CSP で復元されるが，その復元処理は差分データで指定された FP のファイルの読み込みとメモリコピー処理となる．復元されたコンテンツは Fingerprint ヘッダで指定された FP をキーにして FP キャッシュに登録される．また，CSP は復元後のコンテンツを下流に転送する際，SSP で付加された専用ヘッダを削除し退避されていたヘッダを復元して下流に転送する．なお，差分データの配送を HTTP/1.1 の拡張で実現するための手法が文献 27) により提案されているが，我々は本節に示した独自の方式を採用した．

4.6 FP キャッシュのパージと再送処理

SSP と CSP は，双方が持つ FP キャッシュに同じコンテンツが登録されていることを前提に動作する．

しかし FP キャッシュの容量には限りがあるため，双方のプロキシは適宜パージ処理を行い，LRU アルゴリズムに従い古いコンテンツを削除する．その結果，CSP は SSP が FP 転送や差分転送で指定したコンテンツを持っていない状況が起こりうる．その場合には，SSP に対して復元後のコンテンツの FP を指定してコンテンツの再送を依頼する．

再送処理の具体的な処理フローを次に示す．

(1) SSP は FP 転送もしくは差分転送を行う．FP 転送の場合は復元後のコンテンツの FP が Fingerprint ヘッダで指定されている．また，差分転送の場合も復元後のコンテンツの FP が Fingerprint ヘッダで指定されている（4.4 節および 4.5 節参照）．

(2) CSP は SSP から受信した FP 転送もしくは差分転送を受け，オリジナルコンテンツの復元処理を実行する．しかし，復元処理に際して，CSP の FP キャッシュ中に復元に必要なコンテンツがないために復元に失敗する．

(3) CSP は復元後のコンテンツの FP を Fignerprint ヘッダに挿入した GET リクエストを SSP に転送する（再送要求）．

(4) SSP はリクエストヘッダに Fingerprint ヘッダがあることから CSP からの再送要求であると判断し，Fingerprint ヘッダで指定されたコンテンツを FP キャッシュから取得し，そのコンテンツをボディとした HTTP レスポンスを下流に返す．

(5) CSP は該当レスポンスを受信したら FP キャッシュに登録するとともに Fingerprint ヘッダを削除し下流に転送する．

FP 転送や差分転送による再送の発生は，CSP の FP キャッシュの状態を SSP で完全に管理すれば避けることが可能である．しかし，そのためにはキャッシュの状態が変化するたびに双方のプロキシ間に余計なトラフィックが発生し手続きも繁雑になる．また，キャッシュ管理を行っていたとしても通信エラーやタイミングによっては双方のキャッシュに不整合が発生しうる．そのため，本論文では上述のキャッシュ管理を省く代わりに再送処理によってコンテンツの整合性を保証する．

4.7 本手法の特徴

本節では本手法の特徴をまとめる．

(1) 独自の差分方式

コンテンツを行単位でキャッシュし，複数のベースコンテンツによる圧縮が可能．コンテンツを行のハッシュ

値という形で持つことでメモリ使用量を抑制し多数のベースコンテンツを持つことを可能としている．

(2) FP 転送と差分転送

FP 転送により aliasing による HTTP レスポンスの冗長を避けるとともに，差分転送により resource modification や異なる URL でも類似したコンテンツの冗長を避ける．

(3) 再送処理

CSP のキャッシュの状態を管理する代わりに再送処理によってコンテンツの整合性を保証する．

(4) 静的キャッシュ

フィンガープリントキャッシュに URL との対応付けテーブルを保持することで，通常のプロキシキャッシュとしても動作する．

(5) 透過性

Web サーバからみて完全に透過である．またブラウザからみても通常のプロキシとなる．

(6) リバースプロキシ

SSP をリバースプロキシとして配置することが可能である．その場合，ブラウザからみても完全に透過になる．

(7) 設定が不要

適用先サイトに応じた詳細な設定が不要．

(8) HTTP による通信

SSP-CSP 間はすべて HTTP と独自の HTTP ヘッダによって通信が行われ，既存の WWW システムで利用することが可能．

5. 性能評価

本章では我々の方式を用いることで得られるレスポンスデータサイズ削減効果を 3 つの Web サイト（サイボウズ，NIKKEI NET，Yahoo! Japan）で評価した結果と，サイボウズでのデータサイズ削減効果による表示時間の向上効果について評価した結果を示す．

5.1 測定環境

測定を行ったシステム構成を**図 5** に示す．図 5 の社内 LAN/プライベート LAN1/プライベート LAN2 は 100 Mbps のイーサネット LAN である．プライベート LAN1 とプライベート LAN2 は 64 Kbps の ISDN エミュレータを介して接続され，プライベート LAN2 は低速回線をエミュレートする環境とした．また，プライベート LAN1 と社内 LAN は GW を介して接続されている．

表 1 に 図 5 に示されている PC の HW スペックと OS の一覧☆を示す．ただし，Pen-3 とは PentiumIII，Cel とは Celeron，M-Cel とはモバイル Celeron である．また，表にない CSP2 とクライアントは CSP1 と同じスペックの PC を用いた．

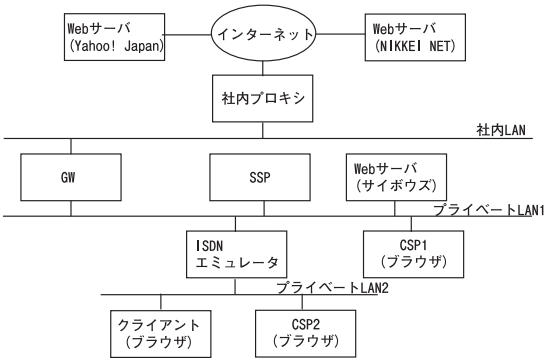

図 5 性能評価システム構成
Fig. 5 System configuration for evaluation of performance.

表 1 HW スペックと OS
Table 1 HW specs and OSs.

	CPU（GHz）	メモリ（MB）	OS
SSP	Cel 1.2	512	Linux 2.4
サイボウズ	Pen-3 1	256	Linux 2.4
CSP1	M-Cel 1.2	128	Windows XP

Web サーバ（サイボウズ）は Apache1.3.26 のアプリケーションとしてプライベート LAN1 上に配置した．また，Web サーバ（NIKKEI NET，http://www.nikkei.co.jp/）と Web サーバ（Yahoo! Japan，http://www.yahoo.co.jp/）は社内プロキシを介してインターネット上にある．SSP は Apache1.3.23 のモジュールとして実装し，プライベート LAN1 上に配置した．SSP はサイボウズへのリクエストは直接転送し，NIKKEI NET および Yahoo! Japan へのリクエストは社内プロキシに転送するように設定した．CSP1 および CSP2 は CSP の機能を Windows のアプリケーションとして実装した．CSP1 および CSP2 のブラウザには IE6（Internet Explorer 6）を用い，プロキシとしてそれぞれのアプリケーションを設定した．また，その上位プロキシとして SSP を設定した．クライアントのブラウザにも IE6 を用い，プロキシは設定しなかった．CSP1 はプライベート LAN1 上に配置した．一方，CSP2 とクライアントはプライベート LAN2 上に配置した．

5.2 転送データ量評価用サイトとアクセスパターン

サイボウズの測定は CSP2 を用いて行った．サイボウズの各ページは 1 つの text/html と装飾用の複

☆ Windows は米国 Micsrosoft Corporation の米国およびその他の国における登録商標または商標です．

表 2 ページ数と URL 数
Table 2 Number of pages and URLs.

	サイボウズ		NIKKEI NET		Yahoo! Japan	
	初回	2 回目	初回	2 回目	初回	2 回目
ページ数	36		64		50	
URL 数	120	40	564	420	1,082	991

数の image/gif から構成されている．評価に際して，トップページ/予定の閲覧/予定の登録/スケジュール（個人月/個人年/個人週/個人日/グループ週/グループ日）/Todo リストの閲覧/Todo リストの登録/My フォルダ/掲示板/ファイル管理/アドレス帳（アドレス一覧/会社一覧/ユーザ名簿）/電話メモ/設備予約/タイムカードからなる 36 ページ（**表 2** 参照）を閲覧・操作した．サイボウズは各ページの HTML が動的コンテンツであり，各ページの類似性が比較的高い場合の代表例として評価を行った．

NIKKEI NET の測定は CSP1 を用いて行った．NIKKEI NET の各ページは text/html と text/css, application/x-javascript, application/ octet-stream, 装飾用の image/gif，広告用の image/jpeg および application/x-shockwave-flash から構成される．ただし，HTTP ボディのない application/x-setcookie は以後の評価では MIME なしとしてカウントする．評価に際して，主要の 13 の記事/企業の 12 の記事/経済の 13 の記事/国際の 13 の記事/社会の 13 の記事からなる合計 64 ページ（表 2 参照）を閲覧した．NIKKEI NET は各ページの HTML が静的コンテンツで，各ページの類似性が非常に高い場合の代表例として評価を行った．

Yahoo! Japan の測定は CSP1 を用いて行った．Yahoo! Japan の各ページは text/html，text/css, text/plain, application/x-javascript，装飾・広告用の image/gif，image/jpeg，application/x-shockwave-flash から構成される．評価に際して，Yahoo! Japan トップページ/ショッピングトップページ/オークショントップページ/チケットトップページ/…/メッセンジャートップページという，Yahoo! Japan トップページとその検索テキストボックスの直下にある 49 のリンクのトップページからなる合計 50 ページ（表 2 参照）を閲覧した．Yahoo! Japan は各ページの HTML が動的コンテンツであり，各ページの類似性が非常に低い場合の代表例として評価を行った．

測定にあたって，SSP と CSP および IE のキャッシュにコンテンツが登録されている場合といない場合を評価するために，上記の 3 サイトをそれぞれ 2 回まったく同じアクセスパターンで閲覧・操作した．なお，サイボウズの評価で初回に登録されたスケジュールと ToDo リストは 2 回目の閲覧・操作を行う前に別のブラウザを用いて削除した．また，差分転送を試みるコンテンツはレスポンスステータスが 2xx で MIME タイプが text/*のものとした．FP 転送を試みるコンテンツはレスポンスステータスが 2xx の全 MIME タイプとした．application/x-javascript はテキストであり差分転送を行うことは可能だが，本章に示す評価では差分対象としなかった．転送データ量は SSP で測定した（5.3 節参照）．

5.3 転送データ量の測定結果

表 2 に閲覧したページ数と転送されたコンテンツの URL 数を示す．サイボウズの初回，2 回目の URL 数はそれぞれ 120，40 であった．NIKKEI NET の初回，2 回目の URL 数はそれぞれ 564，420 であった．Yahoo! Japan の初回，2 回目の URL 数はそれぞれ 1082，991 であった．初回と 2 回目の URL 数の違いは IE のキャッシュにヒットしたため 2 回目では転送が発生しなかったためである．

データ量削減効果について測定結果を**図 6** に示す．図中，“なし” と “あり” とはそれぞれ SSP および CSP がなかった場合とあった場合である．“なし” は SSP が上流から受信した HTTP ヘッダと HTTP ボディの総データ量であり，“あり” は SSP が下流に送信した HTTP ヘッダと HTTP ボディの総データ量である．“なし” はオリジナルの転送データ量に相当し，“あり” は我々の手法を導入した場合の転送データ量に相当する．また，FP 転送とは FP 転送による転送データ量，差分転送とは差分転送による転送データ量，その他とは初めて転送するコンテンツや差分圧縮に失敗したコンテンツ，2xx 以外のレスポンスステータスのコンテンツなどによる転送データ量である．また，3 サイトそれぞれキャッシュが温められていない初回とキャッシュが温められている 2 回目について示してある．

図 6 においてその他の転送は 302 Moved Temporary，差分圧縮に失敗した text/*，初めて転送する image/*や application/*が該当する．いずれのサイトにおいても初回と比較すると 2 回目はその他の転送量が削減されている．これは主に装飾用の image/gif が IE のキャッシュにヒットしたためである．また，いずれのサイトでもなしの場合よりありの場合の方がその他転送量は若干多くなっている．これは初めて転送するコンテンツや差分圧縮に失敗したコンテンツに付加される Fingerprint ヘッダなどの独自ヘッダに起因する．

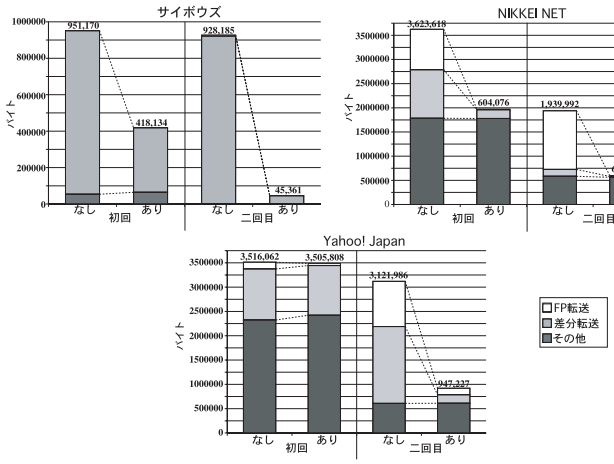

図 6 転送データ量
Fig. 6 Size of data transferred.

なお，本評価では IE 自身が静的なキャッシュ機能を持っているため，SSP や CSP が静的なキャッシュとして動作することはなかった．また，再送も発生しなかった．履歴表は 64K 行とし，古いコンテンツが履歴表で上書きされることはなかった．以下に図 6 に示されている測定結果について説明する．

サイボウズ®☆

サイボウズの初回は，なしの場合が 951,170 バイトに対し，ありの場合は 418,134 バイトとなり元のサイズの約 44.0%に削減された．転送データ量削減は差分転送によって実現され，なしの場合の総転送データ量の約 94.3%がありの場合は約 39.4%のサイズに削減された．

サイボウズの 2 回目は，なしの場合が 928,185 バイトに対し，ありの場合は 45,361 バイトとなり元のサイズの約 4.9%に削減された．転送データ量削減は FP 転送と差分転送によって実現された．FP 転送によってなしの場合の総転送データ量の約 0.7%が，ありの場合は約 2.6%のサイズに削減された．差分転送によってなしの場合の総転送データ量の約 99.1%が，ありの場合は約 4.7%のサイズに削減された．

NIKKEI NET

NIKKEI NET の初回は，なしの場合が 3,623,618 バイトに対し，ありの場合は 2,110,228 バイトとなり元のサイズの約 58.2%に削減された．転送データ量の削減は FP 転送と差分転送によって実現された．FP 転送によってなしの場合の総転送データ量の約 23.1%が，ありの場合は約 1.8%のサイズに削減された．差分転送によってなしの場合の総転送データ量の約 27.7%が，ありの場合は約 17.9%のサイズに削減された．

NIKKEI NET の 2 回目は，なしの場合が 1,939,992 バイトに対し，ありの場合は 604,076 バイトとなり元のサイズの約 31.3%に削減された．転送データ量の削減は FP 転送と差分転送によって実現された．FP 転送によってなしの場合の総転送データ量の約 62.6%が，ありの場合は約 1.5%のサイズに削減された．差分転送によってなしの場合の総転送データ量の約 7.3%が，ありの場合は約 13.8%のサイズに削減された．

Yahoo! Japan

Yahoo! Japan の初回は，なしの場合が 3,516,062 バイトに対し，ありの場合は 3,505,808 バイトとなり元のサイズの約 99.7%に削減された．転送データ量の削減は FP 転送と差分転送によって実現された．FP 転送によってなしの場合の総転送データ量の約 4.0%が，ありの場合は約 29.7%のサイズに削減された．差分転送によってなしの場合の総転送データ量の約 29.9%が，ありの場合は約 97.4%のサイズに削減された．

Yahoo! Japan の 2 回目は，なしの場合の 3,121,986 バイトに対し，ありの場合は 947,227 バイトとなり元のサイズの約 30.3%に削減された．転送データ量の削減は FP 転送と差分転送によって実現された．FP 転送によってなしの場合の総転送データ量の約 29.8%が，ありの場合は約 14.7%のサイズに削減された．差分転送によってなしの場合の総転送データ量の約 50.6%が，ありの場合は約 11.1%のサイズに削減された．

5.4 転送データ量の考察

図 7 に図 6 の FP 転送が行われたものを主要な MIME タイプ別に分類したグラフを示す．差分転送に関しては，NIKKEI NET の初回に 1 回の text/css

☆ サイボウズはサイボウズ株式会社の登録商標です．

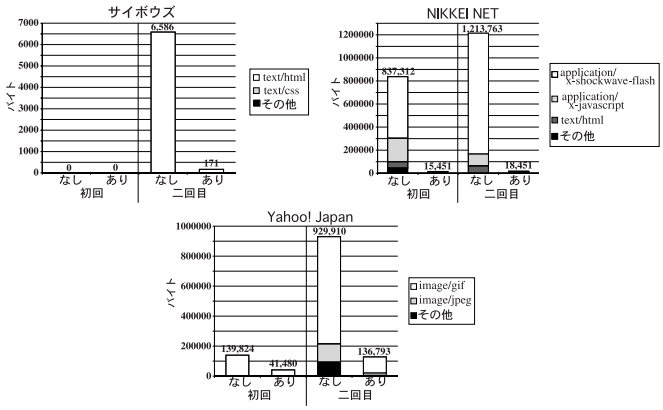

図 7 FP 転送データ量と MIME タイプ
Fig. 7 Size of FP transfers and MIME types.

の圧縮で 34,484 バイトが 26,452 バイトに削減された以外，すべて text/html であった．

サイボウズ

サイボウズの初回は差分転送によって転送データ量が削減された（図 6）．これは，各ページを構成する HTML に多くの共通部分が存在し，それらのコンテンツをベースとして差分圧縮を行った結果である．

サイボウズの 2 回目も主に差分転送によって転送データ量が削減された（図 6）．2 回目と初回を比較すると 2 回目の方が差分転送による削減効果が大きいことが分かる（図 6）．これは，初回にアクセスした同一 URL からの類似性の非常に高い HTML をベースコンテンツにした結果である．

NIKKEI NET

NIKKEI NET の初回は FP 転送と差分転送によって転送データ量が削減された（図 6）．FP 転送は主に広告用のフラッシュと javascript が対象となった（図 7）．差分転送はページ間の類似性が高いため，初回でも圧縮効果が高くなった．

NIKKEI NET の 2 回目は主に FP 転送によって転送データ量が削減された．FP 転送は主に広告用のフラッシュと javascript が対象となった（図 7）．各ページを構成する HTML は静的なコンテンツなため，初回のアクセスで IE のキャッシュに登録され，2 回目では転送が発生しなかった．結果的に 2 回目は差分転送による圧縮効果は低くなった．

Yahoo! Japan

Yahoo! Japan の初回は主に差分転送が行われたが，その圧縮効果は非常に低かった（図 7）．これは各ページの類似性が非常に低いためにキャッシュの温まっていない初回は提案手法では効果が得られなかったことを示している．

Yahoo! Japan の 2 回目は FP 転送と差分転送によって転送データ量が削減された．FP 転送は広告用の GIF/JPEG イメージが対象となった（図 7）．差分転送は初回にアクセスした同一 URL からの類似度の非常に高い HTML をベースコンテンツとした結果，転送データ量が大きく削減された．

5.5 表示時間測定方法

転送データ量削減による表示時間短縮効果を測定するために，図 5 のクライアントと CSP2 を用いて測定を行った．社内プロキシの静的なキャッシュ機能によって後述の 2 回目はキャッシュヒットによって転送時間が削減され正確な測定ができない．そのため，NIKKEI NET および Yahoo! Japan は測定を行わず，社内プロキシを経由しないサイボウズで測定を行った．測定はストップウオッチで行い，リンクのクリックやボタンの押下などの操作を行うと同時に計測を開始し，IE の描画の終了を目視して計測を終了し，その時間を測定値とした．ストップウオッチで計測可能な時間は小数点 2 桁までである．また，本論文で述べたデータ量削減効果による表示時間短縮効果を確認するためにクライアントと CSP2 を用いてそれぞれ測定を行い，SSP と CSP がなかった場合とあった場合の比較を行った．アクセスパターンは 5.1 節に述べたものと同一で 36 ページを閲覧・操作し，キャッシュのない初回とキャッシュの温まった 2 回目の測定を行った．また，測定誤差を軽減するために上記測定を合計 4 セット行った．なお，測定した 4 セットはすべて同じ URL で差分転送や FP 転送，その他の転送が行われた．

5.6 表示時間測定結果

転送データ量と表示時間の測定結果を**表 3** に示す．表 3 のデータ量とは，各セットで転送された総データ量の平均値である．ただし小数点以下は四捨五入し

表 3 表示時間測定結果
Table 3 Result of measuring time to display.

		初回	2 回目
データ量（バイト）	なし	951,067	928,238
	あり	424,122	41,649
表示時間（秒）	なし	157.6 ± 3.6	150.0 ± 3.6
	あり	102.6 ± 3.6	38.4 ± 3.6

てある．また，表 3 の表示時間とは，各ページの表示時間を 5.5 節に示した方法で計測し，それをセットごとに合計した秒数の平均値である．表より，初回は転送データ量が 951,067 バイトから 421,122 バイトに削減されたことにより，表示時間が合計 157.6 ± 3.6 秒から合計 102.6 ± 3.6 秒に削減された．2 回目は転送データ量が 928,238 バイトから 41,649 バイトに削減されたことにより，表示時間が合計 150.0 ± 3.6 秒が合計 38.4 ± 3.6 秒に削減された．5.3 節に示したように，サイボウズでは主に差分転送によって転送データ量が削減される．このため，上記の表示時間短縮は主に差分転送によって得られた効果である．4.4 節および 4.5 節に示したとおり，SSP の処理手順として FP 転送の試行の後に差分転送を試行する．また，差分転送は差分圧縮処理をともなう．そのため，我々の手法は FP 転送のほうが差分転送より処理時間がかからない．FP 転送による転送データ量削減効果を含む場合，同じ圧縮率であるならより表示時間短縮効果が大くなると考えられる．

ここで有効数字と誤差について考察を行う．各計測で誤差の生じる主な要因として，Web サーバの応答時間，ISDN エミュレータを含めたネットワークの状態，IE の描画時間，目視による誤差があげられる．そこで，図 5 のクライアントを用いて測定した場合に着目する．各 36 ページを初回と 2 回目の 2 回ずつ 4 セット測定を行ったため，各ページについて初回と 2 回目それぞれ 4 個の表示時間測定結果がある．初回と 2 回目それぞれ各ページごとに最大の測定値と最小の測定値を求め，その差を算出した．初回は最大 0.45，最小 0.04，平均 0.17 秒の差があった．2 回目は最大 0.29 秒，最小 0.03 秒，平均 0.12 秒の差があった．初回と 2 回目の平均は 0.15 秒である．そこで，Web サーバの応答時間とネットワークの状態と IE の描画，目視による誤差を含めて 0.2 秒を誤差として採用した．差が 0.2 秒に収まる測定値は全体の約 81%である．そこで，各測定データには ±0.1 秒の誤差があるものとし，各測定データの小数点 2 桁目は四捨五入した．

また，セットごとに転送された総データ量は異なったが，その違いは最大で約 400 バイトであり，64 Kbps では約 0.05 秒と換算され無視できる値であった．

6. 拡張と適用先

本論文で提案した手法は，HTTP による通信が行われるため基本的に既存の WWW システムに適用可能である．しかし，サーバ側とクライアント側の双方に専用プロキシを設置する必要がある．そのため運用形態には制限があるが，ASP や無線ネットワークなどその適用範囲は広い．基本的には，SSP と CSP を自前で管理できれば適用可能である．そして，その間のネットワークが低速な場合や従量課金な場合に本技術の効果を得ることができる．上述の本手法の適用やより広域なネットワークへの適用に必要となる SSP/CSP が複数存在するシステムについて 6.1 節で述べる．また，6.2 節および 6.3 節に ASP と無線ネットワークへの適用例を示す．

6.1 複数 SSP/CSP 構成への拡張

これまで SSP と CSP が 1 対 1 の構成で議論してきたが，FP の扱いを拡張することによって SSP が複数ある場合，あるいは CSP が複数ある場合にも対応できる．

(1) 複数の CSP

1 つの SSP に対して複数の CSP が存在する場合は，CSP ごとにキャッシュの内容が異なる．そのため，SSP は CSP が持っていないコンテンツの FP を転送してしまうという問題が発生する．これに対しては，SSP において配信先 CSP 別に FP とコンテンツの対応と履歴表およびハッシュ表を管理することで，未配信の CSP に FP 転送や差分転送を行うという問題を回避できる．SSP がリクエスト元の CSP を判定するために，CSP は 4.5 節で述べた X-FP-Client の値に各 CSP をユニークに識別できる ID を挿入してリクエストヘッダに追加する．

(2) 複数の SSP

1 つの CSP に対して複数の SSP が存在する場合は，複数の SSP が異なるコンテンツに対して同一の FP を発行してしまい，CSP は間違ったコンテンツを受け取るという問題が発生する．これに対しては，FP に各 SSP を識別する情報（IP アドレスなど）を付加することで問題を回避できる．4.2 節で述べた FP の付加的な値とは IP アドレスなどを指す．

上記の組合せで SSP と CSP が多対多の構成である場合にも対応可能である．

6.2 ASP への適用

ASP モデルに基づいて構築されたシステムでは，業務プログラムはオフィスとは別の場所にあるデータセ

ンタ上で運用する．ユーザはオフィスの Web ブラウザからインターネットを介して業務プログラムを利用する．このようなシステム構成において，データセンタの出口に SSP を配置し，オフィスの入口または各 PC に CSP を配置することで本論文で述べた手法を適用することができる．ASP はホスティングだけでなく運用まで含めてコストパフォーマンス良く提供することを目的としている．そのため，経済性の面からデータセンタとオフィス間は LAN に比べて低速なネットワークでつながれている場合が多く，そこがボトルネックになりやすい．さらに ASP の代表サービスである基幹業務プログラムやグループウェアのような情報共有システムは動的コンテンツが多用されているため，本技術の適用によりサービスの高速化が期待できる．

6.3 無線ネットワークへの適用

近年無線ネットワークを介した WWW 閲覧が増加している．無線 LAN など，高速で定額な無線接続の普及も著しいが，従量課金や定額低速な回線での接続も幅広く利用されている．そのような環境下では，無線基地側に SSP を配置し，PC 上に CSP を配置する．それによって従量課金の場合はコストを削減し，定額で低速な回線では応答時間の向上効果が得られる．本技術を利用するユーザは CSP の上位プロキシとして SSP を設定する．一方，本技術を利用しないユーザは従来どおりその無線回線を利用可能である．

7. 結　　論

本論文では，フィンガープリントキャッシュを利用して，同一のコンテンツを再度送ることを抑え，類似のコンテンツは独自の差分圧縮方式による差分転送でデータ量を抑えることができることを明らかにした．

性能評価では，3 サイトでケーススタディを行い，キャッシュがなかった場合とあった場合の転送データ量の測定を行った．その結果，キャッシュが温まっていない状態ではサイトによって我々の方式の効果が得られる場合と得られない場合があることを示した．また，キャッシュが温まっている状態ではいずれのサイトでも我々の方式の効果が得られることを示した．最も効果のある場合は我々の方式を用いることで，転送データ量が元のサイズの 4.9%に削減できることを示した．

また，低速回線下では，転送データ量削減によってユーザが体感するリクエストを出してから画面に表示されるまでの時間を短縮できることを示した．我々の手法を用いることでデータサイズが約 950 KB から 420 KB に削減され，その結果表示時間は約 158 秒から 103 秒に低減された．また，データサイズが約 930 KB から 41 KB に削減され，その結果表示時間は約 150 秒から約 38 秒に低減された．

本論文で述べた方式を使うことで，動的コンテンツを多く使用する場合でも，aliasing と resource modification および異なる URL から類似したコンテンツが転送される冗長を削減することでネットワークのトラフィックを低減し，クライアントからみた応答時間を短縮することができる．

参 考 文 献

1) Luotonen, A.: *Web Proxy Servers*, Prentice-Hall (1997).
2) Housel, B.C. and Lindquist, D.B.: WebExpress: A System for Optimizing Web Browsing in a Wireless Environment, *Proc. 2nd Annual Conference on Mobile Computing and Networking* (*MOBICOM*), pp.108–116 (1996).
3) Berners-Lee, T., Fielding, R. and Frystyk, H.: Hypertext Transfer Protocol — HTTP/1.0, RFC 1945 (1996).
4) Fielding, R.T., Gettys, J., Mogul, J.C., Nielsen, H.F., Masinter, L., Leach, P.J. and Berners-Lee, T.: Hypertext Transfer Protocol — HTTP/1.1, RFC 2616 (1999).
5) Mogul, J., Douglis, F., Feldmann, A. and Krishnamurthy, B.: Potential benefits of delta-encoding and data compression for HTTP, *Proc. ACM SIGCOMM '97*, pp.181–194 (1997).
6) Banga, G., Dougils, F. and Rabinovich, M.: Optimistic Deltas for WWW Latency Reduction, *Proc. 1997 USENIX Technical Conference*, pp.298–303 (1997).
7) van Hoff, A., Giannandrea, J., Hapner, M., Carter, S. and Medin, M.: The HTTP Distribution and Replication Protocol, Submitted to W3C (1997).
8) Psounis, K.: Class-based Delta-encoding: A Scalable Scheme for Caching Dynamic Web Content, *22nd International Conference on Distributed Computing Systems Workshop* (*ICDCSW 2002*) (2002).
9) Rhea, S.C., Liang, K. and Brewer, E.: Value-Based Web Caching, *International World Wide Web Conference 2003* (2003).
10) Hamilton, M., Rousskov, A. and Wessels, D.: *Cache Digest specification, version 5* (1998).
11) Rousskov, A. and Wessels, D.: *Cache Digests* (1998).
12) Kelly, T. and Mogul, J.: Aliasing on the World Wide Web: Prevalence and Performance Implications, *Proc. 11th Intl. WWW Conf.* (2002).

13) Cache Flow Incorperated: Createing a Cache-Friendly Web Site, Cache Flow White Paper (2001).
14) 河内正夫：P2P インターネットの新世紀，社団法人電器通信協会 (2002).
15) 中島　募：EC 時代の Web システム構築，日経 BP 社 (2000).
16) Fox, A. and Brewer, E.A.: Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation, *Proc. 5th International World Wide Web Conference* (1996).
17) Liljeberg, M., Alanko, T., Kojo, M., Laamanen, H. and Raatikainen, K.: Optimizing World-Wide Web for Weakly-Connected Mobile Workstations: An Indirect Approach, *Proc. 2nd International Workshop on Services in Distributed and Networked Environments* (*SDNE*), pp.132–139 (1995).
18) Challenger, J., Dantzig, P. and Iyengar, A.: A Scalable System for Consistently Caching Dynamic Web Data, *Proc. 18th Annual Joint Conference of the IEEE Computer and Communications Societies* (1999).
19) Holmedahl, V., Smith, B. and Yang, T.: Cooperative Caching of Dynamic Content on a Distributed Web Server, *Proc. 7th IEEE International Symposium on High Performance Distributed Computing*, pp.243–250 (1998).
20) Rivest, R.L.: The MD5 Message-Digest Algorithm, RFC 1321 (1992).
21) U.S. Department of Commerce: Secure Hash Standard, FIPS PUB 180-1 (1995).
22) van Hoff, A. and Payne, J.: Generic Diff Format Specification, Submitted to W3C (1997).
23) Rabin, M.O.: Fingerprinting by random polynomials, Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University (1981).
24) Freed, N. and Borenstein, N.: Multipurpose Internet Mail Extention (MIME) Part One: Format of Internet Messages, RFC 2045 (1996).
25) Hunt, J., Vo, K.P. and Tichy, W.: An Empirical Study of Delta Algorithms, *IEEE Soft. Config. and Maint. Workshop* (1996).
26) Mogul, J.C., Douglis, F., Feldmann, A. and Krishmurthy, B.: Potential Benefits of Delta Encoding and Data Compression for HTTP, *Proc. ACM SIGCOMM '97 Symposium*, Cannes, France, pp.181–194 (1997).
27) Mogul, J., Krishnamurthy, B., Feldmann, F., Goland, Y., van Hoff, A. and Hellerstein, D.: Delta encoding in HTTP, RFC 3229 (2002).

(平成 16 年 4 月 5 日受付)
(平成 16 年 12 月 1 日採録)

庄野　篤司（正会員）

1974 年生．1997 年東北大学理学部地球物理学科卒業．1999 年同大学大学院理学研究科博士課程前期課程修了．同年（株）東芝入社．同社研究開発センターにて広域分散システム，ミドルウェアに関する研究に従事．

佐藤　英昭（正会員）

1991 年山梨大学工学部電気工学科卒業．1993 年同大学大学院工学研究科電気工学専攻修士課程修了．同年（株）東芝入社．現在，同社研究開発センターにて広域分散システム，ミドルウェアの研究に従事．

木村　康浩（正会員）

1991 年東京大学工学部電気工学科卒業．1993 年同大学大学院工学系研究科電子工学専攻修士課程修了．同年（株）東芝入社．現在，同社研究開発センターにてミドルウェア，Web システムの研究開発に従事．

木場　雄一

1999 年九州大学工学部情報工学科卒業．2001 年同大学大学院システム情報科学研究科情報工学専攻修士課程修了．同年（株）東芝入社．同社研究所で広域分散システム技術，インターネットに関する研究に従事．

関　　俊文（正会員）

1983 年早稲田大学理工学部電気工学科卒業．1985 年同大学大学院理工学研究科電気工学専攻修士課程修了．同年（株）東芝入社．現在同社研究開発センターコンピュータ・ネットワークラボラトリ所属．博士（工学）．主として分散システムに関する研究開発，およびその応用研究に従事．電子情報通信学会，電気学会各会員．