社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－OS－109 (17)
2008／8／7

P2P 型分散サーバ上のコンテンツ位置推測による一貫性保証

早川 愛　浅原 理人　小島 俊範　河野 健二

慶應義塾大学大学院 理工学研究科 開放環境科学専攻
E-mail: {aiai, asahara, kojima}@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

インターネットにおけるデータ流通量の著しい増加にともない，コンテンツの安定した配信を可能にする基盤として Content Distribution Network (CDN) が利用されるようになった．CDN は，コンテンツの複製をインターネット上に分散配置することで，サーバにかかる負荷を分散させるものである．CDN で扱うサーバの数は増加傾向にあり，Peer-to-peer (P2P) 型のスケーラブルな CDN が提案されている．従来の CDN では特定の管理サーバを用意し，大域的情報を用いて分散配置したコンテンツの一貫性を保証していた．しかし，P2P 型では大域的情報を扱うことはできない．本論文では，管理サーバや大域的情報を必要としない一貫性保証手法として，P2P ネットワーク上で得られる局所的情報のみにより，一貫性を保証する手法を提案する．本手法では，各マシンは隣接したマシンの情報のみを管理すればよく，特定のマシンに負荷が集中しない．コンテンツの提供元サーバで更新があると，局所的な情報のみからコンテンツのある場所を予測し，効率よく更新を通知する．また，優先度の低い Gossip Algorithm による通知を併用し，予測通知を補って全コンテンツを更新する．実験により，提案手法ではコンテンツ配信サーバへの更新通知を Gossip Algorithm のみによる通知より最大 29 % 高速化でき，効率よく一貫性が保証できることを確認した．

Efficient consistency algorithm by speculating the location of the contents on peer-to-peer distributed servers

Ai Hayakawa　Masato Asahara　Toshinori Kojima　Kenji Kono

Department of Information and Computer Science, Keio University
E-mail: {aiai, asahara, kojima}@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

As data traffic on the Internet is dramatically increasing, content distribution networks (CDNs) come to be used to provide stable service. CDNs distribute replicas of contents on the Internet to manage load balance. The number of servers on CDNs is increasing, therefore scalable CDNs using peer-to-peer (P2P) technique is developed. In traditional CDNs, a dedicated server maintains consistency of distributed contents with global information, but in P2P CDNs peers do not have global information. In this paper, we propose an efficient consistency algorithm using only local information without a dedicated server nor global information. With our algorithm, each machine just needs to manage information about neighbor machines. When a machine with original contents is updated, it speculates the location of replicated contents with local information then it propagates update efficiently. To ensure that update messages reach all contents, our algorithm also use low priority message using Gossip algorithm. Experimental results show that our algorithm propagates update messages to all replicated contents faster than simple Gossip algorithm by up to 29 %.

1 はじめに

インターネット上でのデータ流通量の著しい増加にともない，コンテンツを配信するサーバに大きな負荷がかかるようになった．この負荷が大きすぎると，サーバが異常停止してコンテンツの配信ができなくなってしまう．そこで，コンテンツの安定した配信を可能にする基盤として Content Distribution Network (CDN) が利用されるようになった．CDN とは，コンテンツの複製をインターネット上に分散配置し，効率的にコンテンツを配信するネットワークである．クライアントは最寄りのサーバからコンテンツを得ることで，待ち時間を短縮することができる．また，クライアントからのリクエストも分散することができるので，一つのサーバに負荷が集中するのを防ぎ，可用性を高めることができる．近年では Akamai [1], Globule [2] などさまざまな CDN が提案されている．CDN で扱うコンテンツ量およびサーバの数は増加傾向にあり，例えば Akamai では現在 25,000 台以上のサーバを使用している．

このようなサーバ数の増加にともなって各サーバにかかる負荷を軽減させるために，大域的な情報を必要としない CoralCDN [3], ExaPeer [4] のような Peer-to-peer (P2P) 型の CDN が提案されている．このとき，各サーバに分散配置したコンテンツの一貫性保証が必要である．コンテンツが更新された場合，複製されたコンテンツも同様に更新しなければ，ク

ライアントによって異なった内容のコンテンツが配信されてしまう．従来の中央サーバを持ったCDNでは，大域的情報をもったサーバにより分散配置したコンテンツを一元管理していた．しかしこの手法では，管理サーバはコンテンツの種類と複製が配置されている場所をすべて把握しなければならない．P2P型のCDNでは，このような大域的情報を扱わないようにすることで負荷を軽減しているので，従来手法を使うことはできない．P2P型のCDNには，管理サーバや大域的情報を必要とせず，かつ迅速な更新が可能な一貫性保証手法が必要である．

本論文では，P2Pネットワークで得られる局所的な情報のみにより，一貫性保証を行う手法を提案する．提案手法では，各マシンは隣接したマシンの情報のみを管理し，ネットワークの規模が大きくなっても各マシンが管理する情報は増えない．また，管理サーバなどの特定の役割をもつマシンはなく，更新はネットワーク上の物理マシンを経由して通知される．

P2Pネットワークを構成するCDNとしてExaPeer [4]がある．ExaPeerは特定の管理サーバをもたず，クライアントのアクセス経路から需要の分布を把握し，需要の変動に応じて動的にコンテンツの複製を再配置する．需要の変動は，隣接マシンと需要に関する情報を交換することで分析する．本論文では，このExaPeerに一貫性保証手法を適用した．

P2P型のシステムでは，各物理マシンが得られる情報は隣接マシンについての情報のみで，コンテンツのある場所を直接知ることはできない．そこで，局所的な情報のみからコンテンツのある場所を予測し，その方向へ優先的に更新を通知する．提案手法では，P2Pネットワーク上においてリクエストが転送されてくる方角を隣接マシンのリクエスト転送量から算出し，その方向にコンテンツの複製が多数配置されているとみなす．また，予測による通知を補うために優先度の低い通知を併用して，分散配置された全コンテンツを更新する．この通知にはGossip Algorithm [5]を用いる．

実験により，提案手法ではコンテンツをもつマシンへの更新通知がGossip Algorithmのみによる通知より最大29％高速化できることを確認した．また，予測通知と補足通知を組み合わせることで効率よく更新を行いつつすべてのマシンに更新を通知できることを確認した．さらに，Gossip Algorithmのみによる通知と比較して，予測通知を用いるとコンテンツ配信中のマシンすべての更新が完了するのに必要なメッセージ転送数がのべ1106回少なくなり，効率よく通知できることを確認した．

以下，2章ではP2P型でのCDNの一貫性保証に必要な条件を分析する．3章では提案手法を適用したExaPeerについて説明する．4章では提案手法のアルゴリズムを説明する．5章では実験により，提案手法によって更新の通知が効率良く行えることを示す．6章では関連研究を紹介し，最後に7章で本論文をまとめる．

2 P2P型での一貫性保証

P2P型のシステムでは，各マシンが管理する情報が隣接マシンの情報など限られているので，ネットワーク全体のマシン数が増えても性能が低下しない．また，特定の機能をもつマシンも存在せず，各マシンは独立して動作するので一部のマシンが停止してしまってもシステム全体は稼働し続けることができる．サーバが増加傾向にある現在，これらの特徴をもつP2P型のシステムは有用である．

しかしP2P型ネットワーク上の各マシンは大域的な情報をもたないため，一貫性の管理が難しくなっている．従来の中央サーバ型のCDNでは，大域的情報をもった管理サーバが存在するため，この管理サーバによって分散配置したコンテンツを一元管理していた．しかしこの手法では，ネットワーク上のマシンが増加するとサーバが管理すべき情報量も増えてしまう．また更新直後には，管理サーバが複製をもつサーバへ更新の通知をする必要があり，負荷が集中する．この手法では，P2P型の利点が損なわれるので，管理サーバを使わずに一貫性を保証する必要がある．したがって，局所的情報のみからコンテンツの配置情報を得て，迅速に更新を通知できる一貫性保証手法が必要である．

3 ExaPeer

ExaPeerはP2Pネットワークを構成するCDNである．コンテンツを提供するサーバの数や配置を，需要変動に応じて動的に変更することができる．ExaPeer上のマシンは自身のリクエスト転送数を管理する．あるマシンのリクエスト転送数が増加した場合，そのマシン周辺は需要が高い地域であると考えられるため，コンテンツの配信を開始する．この仕組みにより，ExaPeerは需要の高いところへ動的にコンテンツを配置する．本研究では，このExaPeer上で一貫性を保証する．

3.1 基本構造

ExaPeer は Content Addressable Network (CAN) [6] と Global Network Positioning (GNP) [7] を用いて，物理マシン間でオーバーレイネットワークを構成する．まず，コンテンツの検索とリクエストのルーティングには分散ハッシュ検索の一手法である CAN を用いる．CAN によってコンテンツと対応づけられたマシンを提供元サーバとし，リクエストはすべて提供元サーバへ向けて転送される．もし転送経路の途中にコンテンツの複製を配信しているマシンがあった場合は，そこでリクエストを処理し，それ以上リクエストの転送はしない．CAN は物理マシンに座標をあたえ，ネットワークをゾーンと呼ばれる空間に分割する．各物理マシンは自分の座標が含まれるゾーンを管理する．ExaPeer は CAN により物理マシンに与えられる座標に GNP で算出された座標を用いることで，実ネットワークでの位置関係をオーバーレイネットワーク上に反映する．GNP はネットワーク座標システムで，インターネット上の物理マシン間の距離を推定する手法である．インターネット空間を仮想的な d 次元の座標空間として表し，物理マシン間の RTT を仮想空間上のユークリッド距離に近似する．

3.2 コンテンツの分散配置

リクエストを転送する際，各物理マシンはサービス毎にリクエストの転送数を記録する．図 1 のようにゾーン X, Y, Z がある地域で需要が高くなった場合，その地域からのアクセス経路上にある物理マシンのリクエスト転送数が増加する．図 1 では，ゾーン Q にある物理マシンの転送数が多くなっている．転送数が多くなった物理マシンがコンテンツの配信を開始することで，ゾーン X,Y,Z のクライアントのリクエストは，コンテンツ提供元サーバがある場所より近いゾーン Q で処理されるようになる．そこで，ExaPeer 上のマシンは自身のリクエスト転送数が閾値 *upper* より多くなったらコンテンツの配信を開始する．逆に，転送数が閾値 *lower* より少なくなった場合は，コンテンツの配信を終了する．このように ExaPeer はリクエスト転送数に基づいて需要の変動を把握し，それに応じてコンテンツを動的に再配置する．

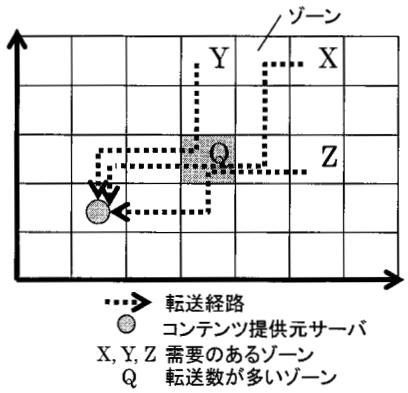

図 1: ExaPeer

4 提案手法

P2P ネットワークで得られる局所的な情報のみにより，一貫性保証を行う．管理サーバを持たないため，更新による負荷は集中しない．また，各物理マシンが管理する情報は隣接マシンについての情報のみで，全体のサーバ数が増えても隣接マシンの数が変わらなければ，各物理マシンが管理する情報量は増えない．コンテンツの提供元サーバで更新があれば，すぐに分散配置されたコンテンツの更新を行うので，クライアントは速やかに最新のコンテンツを得られる．

提案手法では，P2P ネットワーク上においてリクエストが転送されてくる方角を算出し，その方角にコンテンツの複製が多数配置されているとみなす．図 2 のコンテンツ配信中マシン P と需要の関係のように，多くのリクエストが転送されてくる方向は需要が高い地域だといえる．需要の高いところにはコンテンツ提供中のマシンがある可能性が高い．そこで，その方向に優先的な通知 (予測通知) をすることで，効率よくコンテンツに更新を通知し一貫性を保証する．また，予測通知を補うための通知 (補足通知) を併用して，ネットワーク上の全物理マシンに更新が通知されるようにする．補足通知には，大域的情報を必要としない Gossip Algorithm [5] を用いる．Gossip Algorithm は噂を広げるようにして情報を伝えるアルゴリズムで，P2P のような大域的

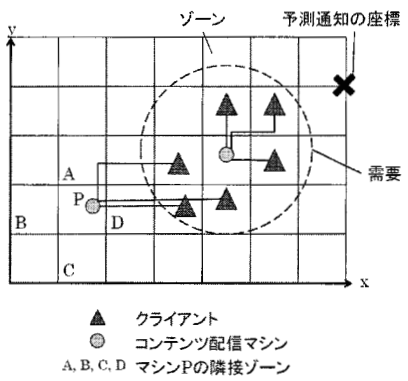

図 2: 予測通知に設定する方向ベクトル

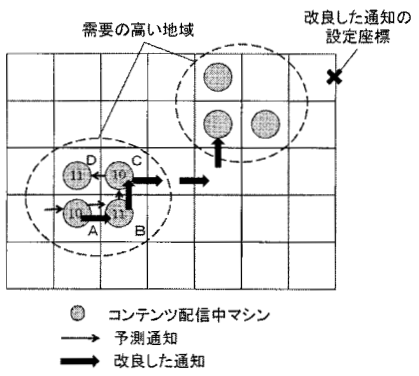

図 3: 予測通知の循環の対策

情報をもたないシステムで使われる．

4.1 予測通知

周辺のリクエスト転送量からコンテンツを配信中の物理マシンがある位置を予測し，その方向に優先的に通知することで効率よく更新の通知を行う．予測の際，コンテンツを配信している物理マシンにはリクエストが集まり，需要の高い地域を予測しやすいと考えられるので，予測通知はコンテンツ配信中の物理マシンで作成する．

まず，コンテンツ配信中のマシンは更新の通知を受け取ると，隣接マシンにリクエスト転送量を問い合わせる．図 2 ではコンテンツ提供中マシン P は，自分が属するゾーンの隣接ゾーン $A \sim D$ に問い合わせる．このとき隣接ゾーン A のリクエスト転送数は 1，ゾーン B, C の転送数は 0，ゾーン D の転送量は 2 である．この場合，予測通知を転送する方向ベクトルは $(x,y) = (x$ 軸方向に隣接する B,D のリクエスト転送量の差，y 軸方向に隣接する A,C のリクエスト転送量の差$) = (2, 1)$ を設定する．

座標を設定した更新通知は CAN のプロトコルに従って転送され，途中のコンテンツ配信サーバに更新を通知する．しかし，コンテンツ配信中のマシンが集中している場合，そのマシンがあるゾーン一帯のリクエスト転送数が総じて多くなっているため，方向ベクトルをうまく定められずに更新通知が一部のマシン間で循環してしまったり，既に通った経路を逆戻りしてしまったりする．

4.1.1 予測通知の循環の対策

通知が循環してしまう例として，図 3 のような状態を考える．通知を受け取ったマシン A は隣接マシン B,D の転送数を比較して座標を設定しなおして B に通知を転送したとする．すると B でも同様に A と C の転送数を比較して座標を設定しなおし C へ転送する．これを繰り返すと，通知はマシン $A \sim D$ の中で循環してしまう．このように，座標を設定した予測通知をコンテンツ配信中のマシンが受け取ったとき，新しく作成した予測通知のみを転送しただけでは，さらに遠方にある需要地域を発見できなくなってしまう．

これを解決するため，既に座標が設定されている通知の場合，まずは設定された座標に通知を転送する．そして改めて座標を設定した通知を作成し転送する．これにより，最初に転送されてきた通知は図 3 の改良した通知として，マシン A,B,C を経由してそのまま転送され，遠方にある需要の高い地域を発見できるようになる．また，マシン B では新しく予測通知が作成されるので，コンテンツの集中している地域ではより多くの予測通知が作成され，迅速にコンテンツの更新ができる．

4.1.2 予測通知の逆行の対策

新しく作成した予測通知が，既に通過してきた需要の高い方へ戻ってしまうことがある．図 4 のような場合，マシン B からマシン C に予測通知が転送され，コンテンツ配信中のマシンである C で新しく予測通知が設定される．マシン C の隣接マシンは A,B,D であり，予測通知を作成すると転送数が一番多い B の方向へ設定され，既に転送されてきた経路を戻ってしまう．この場合も，予測通知の循環と同様に遠方にあるコンテンツに更新を通知するのが遅れてしまい，新しく作成した予測通知も無駄になってしまう．

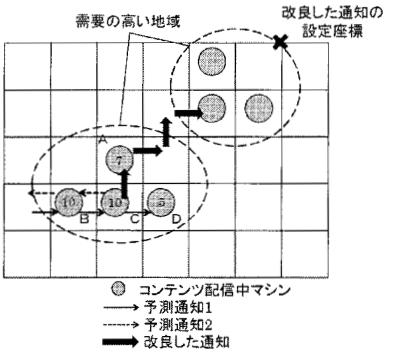

図 4: 予測通知の逆行の対策

これを解決するため，予測通知を作成するためのリクエスト転送数の問い合わせがあったとき，すでに更新通知を受け取ったコンテンツ配信中のマシンは，自分のリクエスト転送数を 0 と申告する．図 4 では，マシン C がコンテンツ位置を推測するためにマシン A,B,D にリクエスト転送数を問い合わせる．マシン B は既に更新の通知を受け取っているので，自分のリクエスト転送数を 0 として申告する．これにより，既に更新された方向にあるリクエスト転送数が考慮されなくなり，予測通知が逆戻りしなくなる．例えば，マシン C ではマシン A,D のリクエスト転送数をもとに予測通知を設定することで図 4 の改良された通知を作成し，遠方にある需要地域まで通知をすることができる．

4.2 補足通知

予測通知によって，現在コンテンツを配信しているマシンに優先的に更新の通知を行うことができる．しかし，更新直後はコンテンツ提供元サーバで作られた通知しかないので，なかなか更新が行われない．そこで，補足通知を用いて提供元サーバの近くにあるコンテンツすべてに更新を通知し，早い段階で十分な数の予測通知を作成できるようにする．これにより，提供元サーバの周りの複数の方向に分散して配置されたコンテンツの方向へ予測通知を転送することができる．また，更新時にはコンテンツを配信していないマシンも，やがてコンテンツの配信を開始する可能性があるので，まだコンテンツの配信を開始していないマシンにも更新を通知しなければならない．そこで補足通知を利用して，全物理マシンに更新を通知する．

補足通知には．大域的情報を必要としない Gossip Algorithm [5] を用いる．まず，提供元サーバは更新があると，隣接マシンへ更新の補足通知を送信する．この補足通知を受け取った物理マシンは，自身が発信元となってさらに自分の隣接マシンに補足通知を転送する．このように，隣接マシンに次々と通知を転送することにより，大域的情報を使わずに全物理マシンに更新を通知することができる．

しかし，単純な Gossip Algorithm による通知ではメッセージ増加にともなってネットワーク全体に負荷がかかってしまう．そこで，補足通知の優先度をさげて負荷を軽減する．補足通知の目的は提供元サーバ周りのコンテンツを更新して十分な数の予測通知をつくることなので，提供元サーバから離れたところにあるコンテンツに到達するのが遅くなってもよい．そこで，優先度を下げるために，予測通知は受信してからすぐに転送をするのに対して，補足通知は受信後に遅延を挟んでから転送する．遅延を挟むことによってメッセージの増加速度を緩めることができる．

5 実験

ネットワークエミュレータ上に提案手法を実装し，更新通知のようすを確認した．ネットワークエミュレータには，大規模分散システムの評価用のエミュレータである Overlay Weaver [8] を用いた．実験では 2 次元空間 上に 2,500 台のノードを配置した．各ノードは 50 x 50 の格子状に配置している．需要の高い地域は図 5 のように座標 A(3000, 3000) を中心とした半径 r = 1500 の範囲と，座標 B(-1000, -1000) を中心とした半径 r = 1500 の範囲に設定した．その中から 0.5 秒ごとにランダムにクライアントを選び，選ばれたクライアントはコンテンツをリクエストする．ExaPeer 上で提供されるコンテンツは 1 種類とし，コンテンツの提供元マシンの位置は (-5000, 5000) とした．用意した仮想ネットワーク上のノードは ExaPeer によって動的にコンテンツの配信を開始したり停止したりする．実験では $upper$ = 160, $lower$ = 5, リクエスト転送数のチェック間隔を 60 秒に設定した．この条件下では 20 個前後のマシンがコンテンツの複製をもち，その配信を開始した．

5.1 予測通知の効果測定

リクエスト転送数に基づいた予測により，コンテンツ配信中のマシンがある方向を見つけて更新の通知ができているかを確認した．補足通知である Gossip Algorithm のみの場合と比較して，予測通知

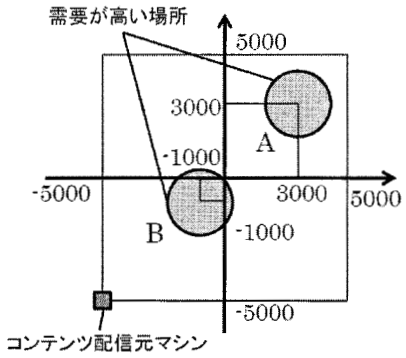

図 5: 物理マシンの配置

によってコンテンツ配信中のマシンへの更新通知が速くなっていれば，コンテンツの位置をうまく予測できているといえる．補足通知では，優先度をさげるために通知を受け取ってから隣接マシンに転送するまでに一定の間隔をおいている．実験では，この間隔を 0.5 秒，1.0 秒，1.5 秒，2.0 秒の 4 種類それぞれに対して行った．本実験は 5 回ずつ行い，その平均を記録した．

図 6 は Gossip Algorithm のみの場合のグラフである．横軸は最初にコンテンツの提供元マシンで更新が行われてからの経過時間であり，縦軸は更新されたコンテンツ配信中のマシンの割合である．Gossip Algorithm のみの場合，更新されたコンテンツ配信中マシンの割合が 30 % ～ 40 % となっているところで，コンテンツ配信中のマシンを更新するのに時間がかかっていることがわかる．これは，Gossip Algorithm では更新を行ったマシンを中心に，更新通知が少しずつ広がっていくだけなので，提供元マシンから離れた位置にあるコンテンツ提供中マシンまで更新を通知するのに，時間がかかってしまうためである．

一方，図 7 は予測通知と補足通知を組み合わせた場合のグラフである．予測通知によってコンテンツ配信中マシンに優先的に通知を行うため，更新されたコンテンツ配信マシンの割合が 30 % ～ 40 % のところでも速やかに更新を通知できている．表 1 に実験結果をまとめる．予測通知と補足通知を組み合わせることで，コンテンツ配信中のマシンの更新は Gossip Algorithm のみによる通知より 0 % ～ 29 % 高速化できることがわかった．遅延の間隔が 1.5 秒のとき，コンテンツ配信中マシンをすべて更新するのにかかった時間は，補足通知のみの場合と予測通知を組み合わせた場合で変わらない．これは，提供元マシンから離れた位置にあるコンテンツにうまく予測通知が到達できなかったためである．しかし，コンテンツ配信中のマシンの 80 % を更新するのに必要な時間を比較すると，予測通知がある場合は 44 秒，補足通知のみでは 56 秒で 12 秒速くなっている．したがって，予測通知によってコンテンツ配信中のマシンへの更新通知の大部分は高速化できていることが確認できた．

表 1: コンテンツ更新にかかった時間 (秒)

間隔	0.5	1.0	1.5	2.0
補足通知のみ	33	52	63	81
予測通知と補足通知の組み合わせ	28	37	63	72
予測通知によって高速化できた時間	5	15	0	9

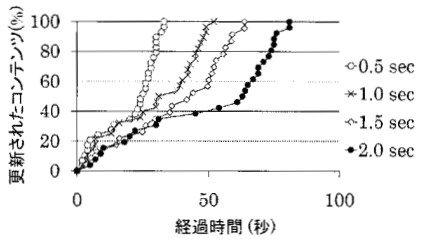

図 6: 補足通知のみでの更新速度

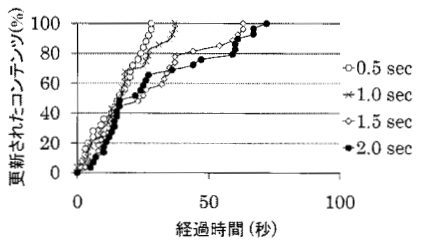

図 7: 予測通知と補足通知の併用した更新速度

5.2 予測通知と補足通知を併用する効果

予測通知と補足通知を併用する効果を検証した．予測通知のみの場合と，予測通知と補足通知を組み合わせた場合で更新の通知ができるコンテンツ配信中のマシンの割合を比較した．

予測通知のみの場合，最初に更新されたコンテンツの提供元マシンで予測通知が作成されてからは，コンテンツの複製を配信しているマシンに到達するまで更新通知が作成されない．更新直後は更新用の

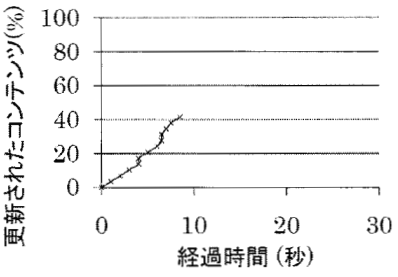

図 8: 予測通知のみでの更新率

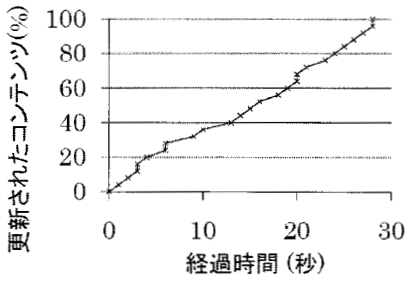

図 9: 予測通知と補足通知の併用した更新率

メッセージが少なすぎて提供元のマシンの近くでコンテンツを配信しているマシンもうまく更新ができていない．その後，少し離れた場所でコンテンツを配信しているマシンに到達して新しく予測通知が作成され，周辺のいくつかのコンテンツ配信中のマシンを更新することができた．しかし，やはり全体のメッセージ数が少ないため，すべてのマシンを更新することはできなかった．図 8 は予測通知のみの場合に更新されたコンテンツ提供中のマシンの割合である．横軸はコンテンツ提供元マシンで更新が行われてからの経過時間，縦軸は更新されたコンテンツ提供中マシンの割合である．予測通知のみでは 41% のマシンしか更新できていない．

一方，予測通知と補足通知を組み合わせた場合の結果を図 9 に示す．補足通知により，コンテンツの提供元マシンの周辺にあるマシンすべてを確実に更新することができている．また，それらの更新されたコンテンツ配信中マシンでさらに遠方にあるコンテンツの位置を予測し，優先的な通知を行うので効率よく全てのコンテンツを更新することができた．また，最終的には全物理マシンに更新を通知することもできた．補足通知を併用することで，予測通知の効果を向上し，かつ全てのマシンに更新を通知することができた．

5.3 メッセージ転送数

予測通知を用いることで，効率良く更新を通知できているかを確認した．補足通知である Gossip Algorithm のみの場合と，予測通知を追加した場合で，コンテンツ配信中のマシンすべてを更新するために必要なメッセージ転送数を比較した．実験は 5 回行い，それを平均した結果を図 10 に示す．横軸は更新されたコンテンツ配信中マシンの数を表し，縦軸は全マシンでのメッセージ転送数の累積である．

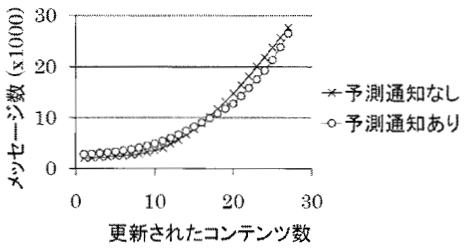

図 10: コンテンツの更新に必要なメッセージ転送数

更新が始まった直後は予測通知がある分，全体に通知するための補足通知のみの場合よりも多くのメッセージが転送されているが，最終的には予測通知がある場合の方がのべ 1106 回少ないメッセージ転送数でコンテンツ配信中のマシン全てに更新を通知することができた．

6 関連研究

ネットワークで扱われるコンテンツやデータの大規模化に伴い，さまざまな CDN が提案されている．この章では，代表的な CDN とその一貫性保証の手法について述べる．

Akamai [1] は代表的な商用 CDN サービスのひとつである．TTL によってコンテンツに有効期限を設けて期限が切れたらキャッシュを削除し，最新のコンテンツを取得することで一貫性を保証する．有効期限が切れないと更新が行われなかったり，逆に有効期限が切れたときは更新がなくてもコンテンツ提供元サーバに接続を行ってしまう．本手法では提供元サーバが更新されるとすぐに更新を通知する方式で速やかに更新を行う．

SPREAD [9] はキャッシュを保存するプロキシ同士で階層構造をつくり，サーバにかかる負荷を分散

させる．各サーバはコンテンツの更新とアクセスの頻度に応じて管理サーバを使う手法と，time to live (TTL) を使う手法を切り替えて一貫性を保証する．どの手法で一貫性を保証するかを決定するため，各プロキシはコンテンツの更新頻度を事前に調べる必要がある．本手法では，各マシンが転送するリクエスト転送量に基づいて一貫性を保証しており，事前にコンテンツの分析をする必要はない．

GlobeDB [10] はアプリケーションが扱うデータベースの複製を各サーバに分散配置するシステムである．データごとに管理サーバを設け，更新があるごとに各サーバに配置したデータベースも更新する．この手法は管理サーバに負荷が集中し，障害が起きると一貫性が保証できない．本手法では，一貫性保証のために特定のサーバを用いないため，負荷が集中することはなく，一部のサーバに障害が起きても他の物理マシンによって一貫性が保証される．

7 おわりに

管理サーバや大域的情報を必要としないスケーラブルな一貫性保証手法として，P2P ネットワーク上で得られる局所的情報のみにより，一貫性を保証する手法を提案した．局所的な情報のみから複製されたコンテンツのある場所を予測し，効率よく更新を通知する．また，予測による通知を補うために優先度の低い Gossip Algorithm による補足通知を併用することで，全物理マシンに更新の通知を行う．

実験により，リクエスト転送量からコンテンツ配信中のマシンがある場所を予測することで，コンテンツ配信中のマシンの更新を単純な Gossip Algorithm による更新の通知より最大 29 % 高速化できることを確認した．また，予測通知と補足通知を組み合わせることで効率よく更新を行いつつすべての物理マシンに更新を通知できることを確認した．さらに，Gossip Algorithm のみの場合と比較して，予測通知を用いるとコンテンツ配信中のマシンの更新に必要なメッセージ転送数はのべ 1106 回少なくなり，効率よく通知が行えていることを確認した．

今後はノードを実ネットワークに近い配置にして実験を行う必要がある．例えば，Transit-Stub トポロジ [11] による実ネットワークを模した仮想ネットワークの構築や，PlanetLab [12] から取得したトポロジ情報を用いることが考えられる．また，アルゴリズムに改良を加えることで，予測通知による更新通知の精度をさらに上げられるとよい．

参考文献

[1] Dilley, J., Maggs, B. M., Parikh, J., Prokop, H., Sitaraman, R. K. and Weihl, W. E.: Globally Distributed Content Delivery., *IEEE Internet Computing*, Vol. 6, No. 5, pp. 50–58 (2002).

[2] Pierre, G. and van Steen, M.: Globule: a Collaborative Content Delivery Network, *IEEE Communications Magazine*, Vol. 44, No. 8, pp. 127–133 (2006).

[3] Freedman, M. J., Freudenthal, E. and Mazières, D.: Democratizing Content Publication with Coral., *Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 239–252 (2004).

[4] Asahara, M., Shimada, A., Yamada, H. and Kono, K.: Finding Candidate Spots for Replica Servers based on Demand Fluctuation, *Proceedings of IEEE International Conference on Parallel and Distributed Systems (ICPADS)* (2007).

[5] Jelasity, M., Montresor, A. and Babaoglu, O.: Gossip-based aggregation in large dynamic networks, *Proceedings of ACM Transactions on Computer Systems (TOCS)*, Vol. 23, pp. 219–252 (2005).

[6] Ratnasamy, S., Francis, P., Handley, M., Karp, R. M. and Shenker, S.: A Scalable Content-Addressable Network., *Proceedings of ACM Special Interest Group on Data Communications (SIGCOMM)*, pp. 161–172 (2001).

[7] Eugene Ng, T. S. and Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches., *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pp. 170–179 (2002).

[8] 首藤一幸, 田中良夫, 関口智嗣: オーバレイ構築ツールキット Overlay Weaver., 情報処理学会論文誌: コンピューティングシステム, Vol. 47, No. SIG12 (ACS 15), pp. 358–367 (2006).

[9] Rodriguez, P. and Sibal, S.: SPREAD: Scalable platform for reliable and efficient automated distribution., *Computer Networks*, Vol. 33, No. 1–6, pp. 33–49 (2000).

[10] Sivasubramanian, S., Alonso, G., Pierre, G. and van Steen, M.: GlobeDB: Autonomic Data Replication for Web Applications, *Proceedings of ACM International World-Wide Web Conference (WWW)*, pp. 33–42 (2005).

[11] Zegura, E. W., Calvert, K. L. and Bhattacharjee, S.: How to Model an Internetwork., *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, Vol. 2, pp. 594–602 (1996).

[12] Bavier, A. C., Bowman, M., Chun, B. N., Culler, D. E., Karlin, S., Muir, S., Peterson, L. L., Roscoe, T., Spalink, T. and Wawrzoniak, M.: Operating Systems Support for Planetary-Scale Network Services., *Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 253–266 (2004).