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

2006－OS－102 (12)
2006／5／13

需要変動に応じたコンテンツの再配置を行う Content Delivery Network

浅原 理人　島田 明男　山田 浩史　河野 健二

慶應義塾大学 理工学部 情報工学科
E-mail: {asamasa, reds, yamada}@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

短時間に数百万ものアクセスが特定のサーバに集中する，Flash Crowd と呼ばれる現象が知られている．Flash Crowd が発生するとサーバは過負荷状態になり，コンテンツを取得するまでのクライアントの待ち時間が大幅に増大する．そのため，Flash Crowd が発生しても安定してコンテンツを提供し続けられるような配信システムが求められている．本論文では Peer-to-Peer 技術を用いた Content Delivery Network "ExaPeer"を提案する．ExaPeer では局所情報のみからコンテンツの需要分布を把握し，需要の高い地域へ優先的にコンテンツの複製を配置する．シミュレーションを行った結果，ExaPeer が Flash Crowd への耐性を備えていることを確認した．

Repositioning Replica in Content Delivery Network based on Demand Fluctuation

Masato Asahara　Akio Shimada　Hiroshi Yamada　Kenji Kono

Department of Information and Computer Science, Keio University
E-mail: {asamasa, reds, yamada}@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

Millions of accesses to a server, called flash crowds, are frequently observed in these days. When flash crowds happen, the server is overloaded and thus the clients must wait for a long time to get contents. In this paper we propose ExaPeer, a content delivery network that can provide contents promptly even in flash crowds. ExaPeer perceives demand fluctuation and repositions replica in the highly demanded areas rapidly.

1 はじめに

インターネットが広く一般に利用されるようになるにつれて，その上で提供されるコンテンツの数や，コンテンツを要求するクライアントの数は増加し続けている．それに伴い，計算機や通信回線の性能の面から弊害が認められるようになってきた．例えば，短時間に数百万ものアクセスが特定のコンテンツに集中する，Flash Crowd と呼ばれる現象が知られている [1]．Flash Crowd が発生すると，コンテンツを提供するサーバは過負荷状態になり，コンテンツを取得するまでのクライアントの待ち時間が大幅に増大してしまう．

こうした問題を解決するため，Content Delivery Network (CDN) という手法が提案されている．CDN ではコンテンツの複製を提供する複数のサーバを地理的に分散配置し，クライアントのアクセスを複数のサーバに振り分ける．このようにすることで，クライアントのアクセスが特定のサーバに集中しなくなり，計算機や通信回線の性能の限界によってクライアントの待ち時間が増大することを防ぐことができる．

本論文では，Flash Crowd にも耐性のある CDN として，Peer-to-Peer 技術を用いた CDN である ExaPeer を提案する．ExaPeer の特徴は以下の通りである．

需要分布の動的把握：各コンテンツについて，どの地域のクライアントからアクセスが多いかを動的に検出する．これによって，Flash Crowd が発生した場合でも，どの地域から多くのアクセスが発生しているのかを把握できる．

コンテンツの動的再配置：各コンテンツに対する需要分布に応じて，コンテンツの複製を動的に再配置することができる．Flash Crowd が発生すると，アクセス数の多い地域に多くの複製を配置することができるため，Flash Crowd の影響を緩和できる．さらに ExaPeer では，大域的な情報を収集することなくこれらの機能を実現する．

ExaPeer はコンテンツを提供するサーバ・ノード

群から成り，個々のサーバ・ノードは Peer-to-Peer 技術を用いて接続されている．ExaPeer では個々のサーバ・ノードが局所情報のみから周辺の需要分布を把握し，需要の高い地域にコンテンツの複製を配置する．そのため，Flash Crowd が発生してもその影響を緩和することができる．

ExaPeer では，分散ハッシュ検索 [2] のひとつである CAN[3] を用いてコンテンツの検索を行う．CAN では各ノードに M 次元の座標を与えるが，この座標に GNP[4] という手法を用いて計算した座標を用いる．GNP を用いて計算した座標は，各ノードのネットワーク上の位置を反映した座標となっており，ネットワーク上で近いノードは，M 次元座標空間上でも近くなるようになっている．

CAN と GNP を組み合わせた構造を用いることにより，需要分布に応じたコンテンツの動的再配置が可能となっている．CAN では，クライアントからの検索要求は，サーバ・ノードをリレーしながら目的のノードに到達する．ある地域のクライアントからの需要が増大すると，特定の経路上のサーバで，検索要求の転送回数が増大する．GNP と組み合わせた構造を用いているため，CAN 上での転送経路はネットワーク上の転送経路に近い．したがって，転送回数の多いノードにコンテンツを配置するだけで，需要分布に応じたコンテンツの再配置が可能となっている．

ExaPeer が Flash Crowd への耐性を備えていることを確認するため，ネットワーク・シミュレータ PeerSim[5] 上でシミュレーションを行った．その結果，コンテンツ取得時のクライアントの平均待ち時間が，シミュレータ時間で約 100 サイクル以内に，Flash Crowd 発生前までの平均待ち時間以下になることを確認した．また，複製の配置を固定した場合よりもクライアントの平均待ち時間が最大約 450 サイクル短かった．

2 基礎技術

ExaPeer ではコンテンツの配置や探索のための構造に CAN を，コンテンツの複製を保持するサーバ・ノードやクライアントの位置関係を，実際のネットワークに対応させるために GNP を用いている．本章では ExaPeer の基本機構に用いた CAN と GNP について述べる．CAN や GNP を知っている場合は本章は読み飛ばして構わない．

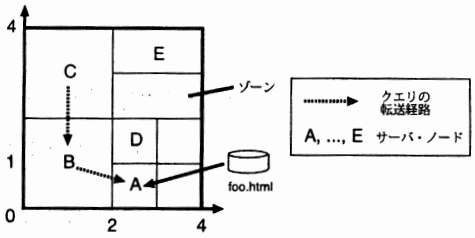


図 1: Content-Addressable Network

2.1 Content-Addressable Network

Content-Addressable Network (CAN) は分散ハッシュ検索の手法のひとつである．CAN では，各サーバ・ノードおよびコンテンツに M 次元の座標を与え，その座標を用いてコンテンツの検索を行う．

CAN では，M 次元の空間をゾーンと呼ばれる領域に分割し，各ゾーンにはひとつのサーバ・ノードが存在するようにしている．図 1 に 2 次元の空間をゾーン に分割した例を示す．あるゾーン X のサーバ・ノードは，その座標がゾーン X 内にあるようなコンテンツを保持する．

クライアントが送信したクエリは，コンテンツのあるゾーンに近づくように，隣接したゾーンに転送する．ゾーンの座標とコンテンツを示す座標とを比較することで，隣接したゾーンの座標からコンテンツのあるゾーンに近づくものを選択する．

図 1 を用いてクエリが転送される様子を説明する．例えばコンテンツ foo.html に与えられた座標が (2.5, 0.5) であった場合，ゾーン A に配置される．クライアントのクエリがゾーン C に対応するサーバ・ノードに送信されたとすると，ゾーン C に対応するサーバ・ノードは隣接するゾーンから，(2.5, 0.5) に近づくゾーン B を選択する．ゾーン B に対応するサーバ・ノードは隣接するゾーン A が (2.5, 0.5) を含むので，ゾーン A に対応するサーバ・ノードにクエリを転送する．

2.2 Global Network Positioning

Global Network Positioning (GNP) は，インターネット上のノード間の距離を推定する手法のひとつである．各ノードには M 次元の座標を与え，その座標から決まるユークリッド距離がインターネット上の距離の近似となるようにする．これによって，任

意のノード間の距離を容易に計算できるようになっている．

GNPでは，あらかじめランドマークとなるいくつかのノードを選択しておく．これらのランドマークにも座標を与え，ランドマーク間の距離がランドマーク間のRTTの実測値に近くなるようにする．具体的には次のようにしてランドマークの座標を定める．ここでは3次元の座標を与える場合を例として説明する．ランドマーク L_1, L_2, L_3, L_4 の座標は次のように求める．座標から計算したランドマーク間の距離と，RTTの実測値が近くなるよう，

$$f_1 = \sum_{L_i, L_j \in \{L_1, \ldots, L_4\}, i>j} (d_{L_iL_j} - \hat{d}_{L_iL_j})^2$$

を最小化するように座標を定める．ここで $d_{L_iL_j}$ は L_i, L_j 間のRTT，$\hat{d}_{L_iL_j}$ は定めた座標から求められる L_i, L_j 間の距離である．

ランドマーク以外のノード A の座標は次のように求める．各ランドマークまでの座標から計算した距離と，RTTの実測値が近くなるよう，

$$f_2 = \sum_{L_i \in \{L_1, \ldots, L_4\}} (d_{AL_i} - \hat{d}_{AL_i})^2$$

を最小化するように座標を定める．ここで d_{AL_i} は A, L_i 間のRTT，$\hat{d}_{AL_i}$ は定めた座標から求められる A, L_i 間の距離である．

文献[6]によると，座標の次元数は計算量と精度のバランスから6次元程度がよく，全体の90%のノードでRTTの実測値との相対誤差が2.0以下となる．

3 ExaPeer

ExaPeerはコンテンツに対するネットワーク全体の需要分布を把握し，適切に複製の再配置を行うことで，Flash Crowdに対処している．コンテンツに対する需要分布を把握するひとつの方法は，コンテンツを要求するクライアントの位置やアクセス頻度を集中管理する方法である．コンテンツを提供する各サーバ・ノードがアクセス頻度等を収集し，複製の最適な配置を決定する．この手法を用いると，原理上，最適な複製の配置を求めることができる．

しかし，この手法はコンテンツ数やクライアント数の増加に対してスケーラブルでない．なぜなら，コンテンツ数やクライアント数の増加に伴って管理する情報量も線形に増加するからである．また，この手法ではFlash Crowdの発生を適切に把握できない場合がある．Flash Crowdは突発的に発生するため，アクセス頻度等の収集は短い間隔で頻繁に行わなければならない．しかし，アクセス頻度等の収集は，ネットワークやサーバ・ノードの負荷を増大させることから，ある程度の間隔をあける必要があり，結果としてFlash Crowdの発生を見逃す場合がある．

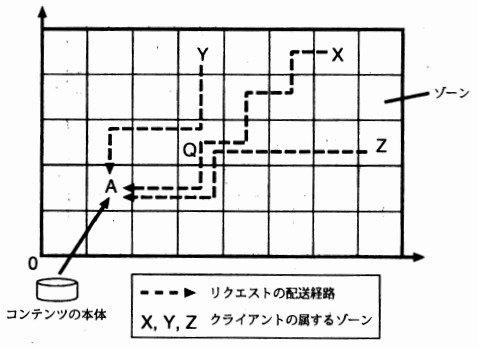


図2: 配送経路による需要分布の把握

本論文で提案するExaPeerでは，個々のサーバ・ノードが局所情報のみを用いて周辺の需要分布を把握し，複製の再配置を実現する．

3.1 基本構造

ExaPeerでは各サーバ・ノードにGNPを用いて計算した座標を与え，ネットワーク上の位置関係を近似している．このGNPを用いて計算した座標からCANの構造を構築する．具体的には，各サーバ・ノードがGNP座標から担当すべきゾーンを決定し，そのゾーンに対応するコンテンツを保持する．

ExaPeerはCANと同様の手法でリクエストの転送を行う．ただし，CANとは異なり，クライアントは必ず自分自身が属しているゾーンのサーバ・ノードからリクエストを送信する．

3.2 配送経路による需要分布の把握

ExaPeerではリクエストの配送経路から需要分布を把握する．サーバ・ノードはコンテンツ毎に，自分が転送したリクエストの数を記録しておく．ある地域からの需要が高まると，その地域からのリクエストの配送経路上にあるサーバ・ノードは転送数が増加する．リクエストの転送数があらかじめ設定し

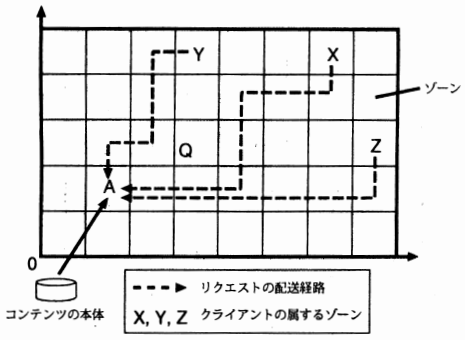


図 3: 需要分布の把握が失敗する例

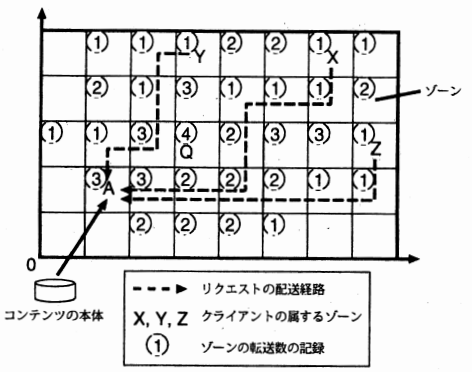


図 4: 配送経路に隣接するゾーンに転送数を加算する

ておいた閾値を超えると，サーバ・ノードは自分の周辺の需要が高まったと判断する．

図 2 に配送経路から需要分布を把握する例を示す．破線の矢印はリクエストの配送経路を示す．この例において，配送経路の集中しているゾーン Q のサーバ・ノードの転送数は 3 であり，配送経路の集中していない他のゾーンのサーバ・ノードは 3 未満である．仮に閾値を 3 とすればゾーン Q の周辺の需要が高くなったとを判断できる．

3.3 配送経路から需要分布を把握する工夫

3.2 節で示した方式では，配送経路からの需要分布の把握が失敗する場合がある．図 3 は需要分布の把握が失敗する例である．図 2 と異なり，3 つのリクエストの配送経路すべてが集中しているゾーンがない．図 2 と同じ需要分布であるにもかかわらず，どのサーバ・ノードも需要の高まりを検出できない．

このような状態になる原因は 2 つある．1 つは，需要の高い地域がコンテンツの本体のあるゾーンから遠いために，リクエストの配送経路に揺らぎが生じてしまうためである．もう 1 つは，そもそも需要のある地域が散らばっていて配送経路の集中する点がないためである．

リクエストの配送経路の揺らぎによる弊害を低減させるために，ExaPeer では配送経路に隣接するゾーンにも，リクエストの転送を行ったとみなして転送数を加算する．図 3 においてゾーン X からの配送経路はゾーン Q を含んでいない．ここで配送経路に隣接するゾーンにも転送数を加算すると，ゾーン Q にも転送数が加算されるので，ゾーン Q が配送経路に含まれているとみなすことができる．このようにして配送経路に広がりを持たせ，配送経路の重なりが現れやすいようにしている．なお，配送経路の重なりという特徴をよく表せるように，隣接するゾーンに加算する転送数と経路上のゾーンに加算する転送数は区別せず，どちらも実際に転送が起こったとみなす．

図 4 は，図 3 で経路に隣接するサーバ・ノードにも転送数を加算したときの例である．円中の数字が，加算した分も含めた各ゾーンの転送量を表している．この例ではゾーン Q は 4 つのメッセージを転送したことになり，ゾーン Q のサーバ・ノードは図 2 と同じく閾値を 3 以上とすることで，周辺の需要が高いと判断できる．

また，需要のある地域が散らばっている場合に対処するために，ExaPeer ではコンテンツの周辺において需要のある地域の方向に位置するサーバ・ノードに複製が配置されやすくしている．具体的には，コンテンツを持つサーバ・ノードへリクエストを転送したときは，加算する転送数を n 倍する．例えば図 3 では需要のあるゾーン X, Y, Z はゾーン A からみて右上の地域にある．リクエストの転送規則から，ゾーン X, Y, Z からのリクエストは必ずゾーン A の右方向もしくは上方向からゾーン A に到達する．よって，コンテンツの本体を持つゾーン A へ転送したサーバ・ノードが記録する転送数を n 倍することで複製が配置されやすくなる．このようにすることで，コンテンツの複製を持つサーバ・ノードの位置が徐々に需要のある地域へ近づいていくことが期待できる．

図 5 は，図 4 の例でコンテンツを持つサーバ・ノードへ転送した際は処理数を 2 倍とした場合を示す．

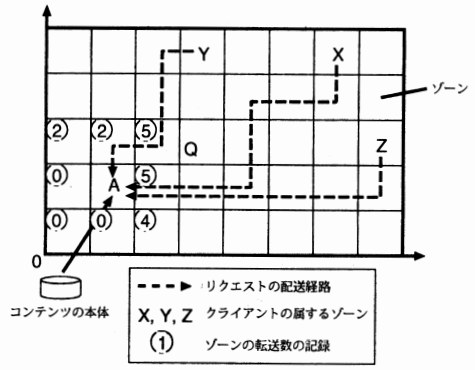


図 5: コンテンツのあるサーバ・ノードへ転送した際は 2 倍とする

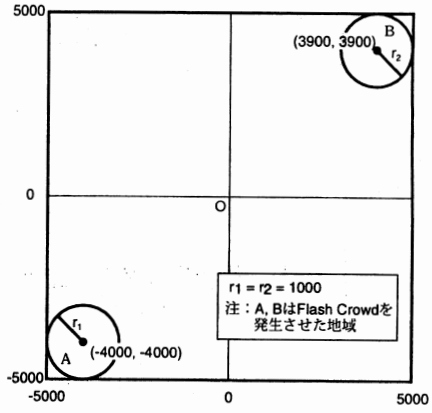


図 6: 実験用仮想 ExaPeer ネットワーク

コンテンツのあるゾーン A の周辺にあるゾーンのサーバ・ノードのうち，要求メッセージの転送経路の起点であるゾーン X, Y, Z の方向にあるゾーンのサーバ・ノードの転送数はそれぞれ 5, 5, 4 である．一方，X, Y, Z の方向以外であるゾーンのサーバ・ノードの転送数は 2, 2, 0, 0, 0 である．よって，要求メッセージの転送経路の起点に近いサーバ・ノードの転送数の記録が大きくなっていることがわかる．現在の ExaPeer では，この例と同様に $n = 2$ としている．

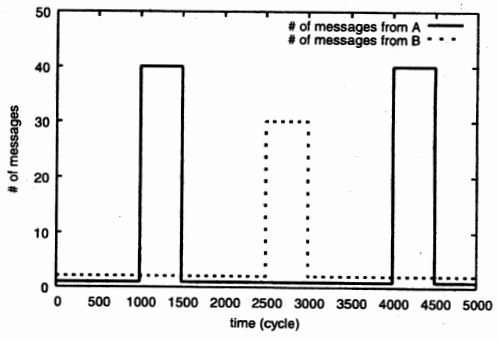


図 7: 地域 A，B からのアクセス数の推移

3.4 コンテンツの動的再配置

ExaPeer は以上で述べた配送経路による需要変動の把握法に基づいて，コンテンツを動的に再配置する．ここではその方法について述べる．

サーバ・ノードは 3.2 および 3.3 節で述べた方式に沿って，コンテンツ毎にリクエストの転送数を記録しておく．この記録は過去一定時間の転送回数を記録するウィンドウになっている．転送数があらかじめ設定しておいた閾値 *UPPER* を上回ると，サーバ・ノードはコンテンツのリクエストを送信し，クライアントと同様にコンテンツを取得する．この取得したコンテンツを複製として提供する．転送数があらかじめ設定しておいた閾値 *LOWER* を下回ると，サーバ・ノードは保持しているコンテンツの複製を削除し，コンテンツの提供を終了する．

複製を持つ条件となる閾値 *UPPER* と，消去する条件となる閾値 *LOWER* はコンテンツ毎に設定できる．*UPPER* が低いと配置される複製の数は増加し，高いと減少する．*LOWER* が低いと配置された複製は短時間で消去され，高いと複製が保持される期間が長くなる．

4 実験

4.1 実験環境

ExaPeer をネットワーク・シミュレータ PeerSim 上に実装し，Flash Crowd に対する耐性の評価を行った．PeerSim は分散アルゴリズムを評価するシミュレータであり，ネットワークの遅延時間や通信帯域などは考慮されていないが，ホストが数万台以上の規模のネットワークをシミュレートする能力がある．本論文では ExaPeer が数万台の規模のネットワークで正常に動作し，Flash Crowd への耐性を備えているか確認するために PeerSim を用いた．

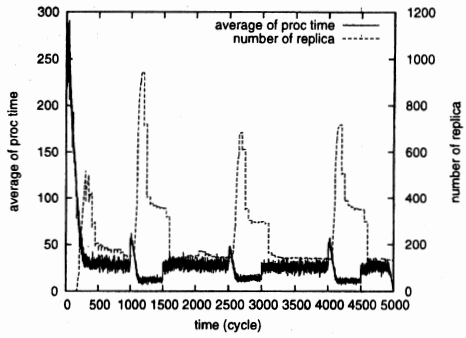


図 8: 受信に要した時間の平均と複製の数の推移

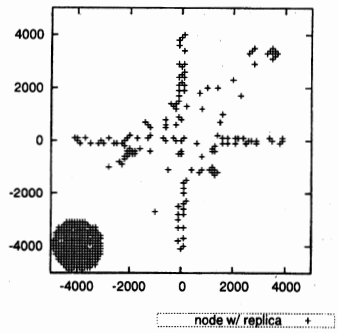


図 9: 複製の配置（1260 サイクル経過時）

図 6 は実験の際に用意した仮想 ExaPeer ネットワークを示す．仮想 ExaPeer ネットワークの GNP 座標の次元は 2 とし，図 6 の示す領域中に，一辺の長さが 100 であるゾーンを 100 × 100 の 10000 個，格子状に配置した．各ゾーンには 1 つずつサーバ・ノードを割り当て，コンテンツの本体は図 6 の原点を含むゾーンにあるとした．各サーバ・ノードは PeerSim のシミュレーション時間の 1 サイクル毎に，クライアントからのコンテンツ要求メッセージを 1 ずつ処理を行い，50 サイクル毎に，コンテンツの動的再配置の判定を行う設定とした．また，転送数を記録するウィンドウのサイズを，過去 100 サイクル分とした．

図 6 において，中心が (-4000, 4000)，半径が 1000 の領域を地域 A，中心が (3900, 3900)，半径が 1000 の領域を地域 B とする．地域 A および地域 B から Flash Crowd が発生したと仮定して，この仮想 ExaPeer ネットワークに次のような負荷を与えた．まず Flash Crowd による負荷を想定して，地域 A および B から図 7 の示すような数のサーバ・ノードをランダムに選択し，各サーバ・ノードがリクエストをひとつ送信する．そのほかに定常的なクライアントのアクセスを想定して，1 サイクル毎にランダムに 30 のサーバ・ノードを全サーバ・ノードから選択し，各サーバ・ノードがリクエストをひとつ送信する．

4.2 実験結果

図 8 は，コンテンツを受信するまでのクライアントの平均待ち時間と，配置した複製の数の推移を示す．Flash Crowd が発生した 1000，2500，4000 サイクルの直後はクライアントの平均待ち時間が 30 前後から最大 60 程度まで増加するが，それに合わせて複製の数も 150 前後から最大 940 まで増加している．その後約 100 サイクル以内に，クライアントの平均待ち時間が 13 前後と，Flash Crowd の発生前以下の水準にまで減少している．

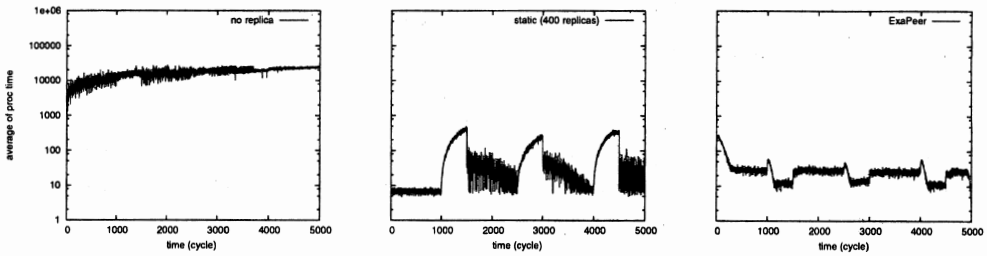
例えば 1260 サイクルの時点では，図 9 が示すように Flash Crowd の発生している地域 A に複製が集中している．これにより，Flash Crowd の発生している地域のサーバ・ノードが自身の周辺の需要が高まっていることを検出し，複製を配置していることがわかる．

図 8 は，Flash Crowd 発生後は発生前より，クライアントの平均待ち時間が小さくなっていることを示している．具体的には，Flash Crowd 発生前が平均 30 程度であったのに対して，Flash Crowd 発生後は平均 13 程度で安定している．これは Flash Crowd の発生地域が非常に大きな負荷を生み出しているために，他の地域よりも複製の配置が密になったためである．

また図 8 によると，Flash Crowd の発生直後は非常に多くの複製が配置されているが，しばらくすると地域 A の Flash Crowd では 380 前後，地域 B の Flash Crowd では 300 前後で安定している．これは ExaPeer が Flash Crowd の規模や分布を把握して，コンテンツの再配置が行えていることを示す．

図 10 は複製の配置を，1) 配置しない，2) 固定して配置する，3)ExaPeer の方式で配置する，とした場合の，クライアントがコンテンツを取得するまでに要する平均時間の推移を示したものである．図 10 の縦軸は対数である．固定配置は，400 個の複製を格子状に 500 間隔で配置した．

複製を配置しない方式では，クライアントの平均



a) 複製を配置しない　　b) 固定配置　　c) ExaPeer

図 10: 地域 A，B からのアクセス数の推移

待ち時間がシミュレーション開始時点から 10000 サイクル以上であり，シミュレーションが進むにつれて増加している．

400 個の複製を固定して配置した方式は，最初の 1000 サイクル間はこの 3 方式の中で最もよく，7 前後の値をを示している．しかし Flash Crowd が発生すると，Flash Crowd の発生している間はクライアントの待ち時間が増加し続けている．また Flash Crowd が終わった後は待ち時間の平均値が 10 から 100 の間で揺らぎがある．これは，Flash Crowd が発生した地域の複製を持つサーバ・ノードに，未処理のリクエストが残っているからである．

ExaPeer の方式は Flash Crowd 発生時以外の平均待ち時間は 30 前後であり，Flash Crowd が発生しても約 100 サイクル後には Flash Crowd 発生前以下の値まで下がっている．また Flash Crowd の発生後に待ち時間の揺らぎが大きくなることもない．複製を固定して配置する方式と比較して最大約 450 サイクル短かった．

5 関連研究

現在もっとも広く使われている商用 CDN に Akamai[7] がある．Akamai は EdgeServer と呼ぶキャッシュ・サーバと，クライアントの近傍にある EdgeServer の IP アドレスを返すような独自の DNS サーバを持つ．クライアントは最寄りの EdgeServer を経由してコンテンツを取得し，その際 EdgeServer はコンテンツをキャッシュする．Akamai では Flash Crowd が発生すると，十分な数の EdgeServer にキャッシュが生成されるまで，コンテンツの提供元のサーバにアクセスが集中する．そのため，Flash Crowd 発生直後はコンテンツの提供元であるサーバの負荷が高くなる．それに対して ExaPeer では，既に複製を持つサーバ・ノードから取得することができるので，コンテンツの提供元のサーバ・ノードの負荷を抑えられる．

SPREAD[8] はプロキシ・サーバを階層的に接続し，クライアントからのアクセスをコンテンツの本体を持つサーバまでプロキシ・サーバ間で転送していく．プロキシ・サーバの関係を階層的な構造にすることで，Flash Crowd の発生直後も，コンテンツの本体を持つサーバに集中する負荷を軽減している．Flash Crowd 発生時は，Flash Crowd の発生地域からコンテンツの提供元であるサーバまでの経路にある，すべてのプロキシ・サーバにコンテンツがキャッシュされる．そのため，Flash Crowd の発生地域とは異なる地域にまでキャッシュが生成されてしまい，ハードウェア資源を浪費する．ExaPeer は需要変動に応じて複製を再配置するので，Flash Crowd の発生地域のみへ効率的に複製を配置できる．

RaDaR[9] はネットワーク層のルーティング情報からクライアントのアクセス経路を割り出し，アクセスが集中している点に近いサーバにコンテンツの複製を配置する CDN である．RaDaR では複製の配置情報を一元管理しており，配置が変更されると配置情報を変更する必要がある．Flash Crowd が発生すると，RaDaR では Flash Crowd の発生地域のサーバにコンテンツの複製を配置するとともに，一元管理している配置情報を更新する．よって，Flash Crowd の規模が大きいと複製の数が多くなるため，配置情報の更新の手間も増大する．

Peer-to-Peer 型の CDN に，Globule[10]，CoralCDN[11] がある．Globule はクライアントのアクセス状況からシミュレーションを行い，コンテンツの複製の最適な数と位置を算出し複製を配置する．Globule は SCoLE[6] という，ネットワーク距離から絶対座標空間を構築する機構を持ち，クライアント

の位置と数から需要分布を把握し，複製を配置する機構を備える [12]．Globule はクライアントのアクセス情報を収集し，サーバでシミュレーションを行って配置を決定する．シミュレーションを行わない限り複製の配置は変更されないので，Flash Crowd という突発的な需要変動に対応するためには，シミュレーションの間隔を Flash Crowd に耐えられるように十分小さくする必要がある．

CoralCDN はインターネット上に分散配置された，コンテンツの複製をキャッシュするノード群を経由してコンテンツの本体を持つサーバにアクセスし，ノード群がコンテンツをキャッシュする CDN である．クライアントのアクセスを受けたノードは，分散ハッシュ検索を用いて CoralCDN のノード群からキャッシュを検索し取得する．Flash Crowd が発生すると発生した地域のノード群にキャッシュが生成され負荷を分散する．CoralCDN は需要の規模にかかわらずクライアントのアクセスを受けたノードがキャッシュを生成する．そのため需要が低い場合はキャッシュが過剰に配置され，ハードウェア資源を浪費する．ExaPeer は需要の規模に応じてコンテンツの複製の配置を調整することができる．

6 おわりに

本論文では Flash Crowd への耐性を持つ Peer-to-Peer 型 CDN である ExaPeer を提案した．ExaPeer はクライアントからコンテンツの本体を持つサーバ・ノードまでの通信経路を，CAN 上でのリクエストの配送経路という形で表し，その分布からコンテンツの需要分布を把握する．配送経路の分布は，個々のゾーンが持つ，リクエストの転送回数という局所情報から推測する．この転送回数の大小から複製の配置を判断する．シミュレーションの結果，コンテンツ取得時のクライアントの平均待ち時間が，シミュレータ時間で約 100 サイクル以内に，Flash Crowd 発生前までの平均待ち時間以下になることを確認した．また Flash Crowd に対して，複製の配置を固定した場合よりもクライアントの平均待ち時間が短かった．

今後は，ネットワークの通信遅延時間や帯域幅といった条件を考慮したネットワーク・シミュレータ上での検証を行う必要がある．また，実際のインターネット上のトラフィック情報を元にしたシミュレーションによる ExaPeer の有効性の確認や，他の CDN との比較実験を行う必要があると考えられる．

謝辞

本研究の一部は，科学技術振興機構 CREST「ディペンダブル情報処理基盤」による支援を受けている．

参考文献

[1] Lorenz, S.: Is your Web site ready for the flash crowd? (2000). http://www.serverworldmagazine.com/sunserver/2000/11/flash.shtml.

[2] Balakrishnan, H., Kaashoek, M. F., Karger, D. R., Morris, R. and Stoica, I.: Looking up data in P2P systems., *Communications of the ACM*, Vol. 46, No. 2, pp. 43–48 (2003).

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

[4] Eugene Ng, T. S. and Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches., *INFOCOM*, Vol. 1, pp. 170–179 (2002).

[5] *PeerSim*. http://peersim.sourceforge.net/.

[6] Szymaniak, M., Pierre, G. and van Steen, M.: Scalable Cooperative Latency Estimation., *International Conference on Parallel and Distributed Systems*, IEEE Computer Society, pp. 367–376 (2004).

[7] 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).

[8] 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).

[9] Rabinovich, M. and Aggarwal, A.: RaDaR: A Scalable Architecture for a Global Web Hosting Service., *Computer Networks*, Vol. 31, No. 11-16, pp. 1545–1561 (1999).

[10] Pierre, G. and van Steen, M.: Globule: A Platform for Self-Replicating Web Documents., *Protocols for Multimedia Systems*, Lecture Notes in Computer Science, Vol. 2213, Springer, pp. 1–11 (2001).

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

[12] Szymaniak, M., Pierre, G. and van Steen, M.: Latency-Driven Replica Placement., *Symposium on Applications and the Internet*, IEEE Computer Society, pp. 399–405 (2005).