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

2007－OS－105 (17)
2007／4／6

需要変動に応じた動的再配置が可能なミラーサーバ基盤

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

インターネットの利用者数は年々増加傾向にあり，コンテンツの配信にかかる負荷も増大している．現在ではインターネットの信頼性の低さから発生する現象もしばしばみられる．例えば Flash Crowds と呼ばれる，短期間に不特定多数のアクセスが特定のサービスを提供するサーバに集中する現象がある．クライアントの負荷を分散するために複数のミラーサーバを配置することがあるが，この配置は静的に行われるため適切な配置が難しい．本論文では，Flash Crowds に対応した動的再配置可能なミラーサーバ基盤として，ExaPeer を提案する．ExaPeer 上では，サービス提供者が提供するサービスの需要の規模や分布を事前に把握する必要はない．ExaPeer はオーバレイネットワークを構築し，サービスの需要変動に応じて，サービスを提供するミラーサーバの数や位置を動的に変更する．シミュレーションを行った結果，サービスを受けるまでにかかる遅延時間の平均値が元のサーバの位置より平均 82 %減少する位置に，ミラーサーバを再配置することを確認した．

Repositioning Replica-Servers 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

The number of the Internet users and the loads to deliver contents are growing each year. There is a phenomenon caused by low dependability of the Internet. For example, a flash crowd is unpredictable demand increase on the servers which provide services. Service providers often locate some replica-servers to distribute loads but appropriate locating of replica-servers is difficult because this locating is static. In this paper we propose ExaPeer, an infrastructure for repositioning replica-servers to handle flash crowds. On ExaPeer, service providers need not understand the scale and distribution of the demand for a service. ExaPeer constructs an overlay network of physical machines and changes the number of replica-servers and repositions the location of replica-servers dynamically according to demand fluctuation of a service. Simulational results demonstrate ExaPeer locates replica-servers at the positions where the average time to get a service is decreased by 82 % compared with the original position.

1 はじめに

インターネットの利用者数は年々増加傾向にある．また，現在では動画や楽曲といった大規模コンテンツや，リクエストに応じて動的に生成されるコンテンツの提供も一般的なものとなっている．こうした変化に対して，現在のインターネットの信頼性は十分に応えられるものではない．たとえば Flash Crowds [1] と呼ばれる，短期間に不特定多数のアクセスが特定のサービスを提供するサーバに集中する現象がある．Flash Crowds が発生するとサービスを提供するサーバは負荷が増大し，最悪の場合過負荷状態になってサービスの提供が満足に行えなくなったり異常停止したりする．

高負荷時の信頼性を高めるために，複数のミラーサーバを配置しクライアントからの負荷を分散することがある．この方式ではミラーサーバの配置が静的に決まるため，負荷が一定であるようなサービスに対して有効な方法である．

しかし，ミラーサーバを静的に配置する方法では，需要が動的に変動するサービスに対しては対処することが難しい．このようなサービスでは通常，負荷がもっとも高かったときの状態を元にして，ミラーサーバの位置や数を静的に決定する．これには，1) 通常の負荷ではミラーサーバの数が多すぎる，2) 想定した規模や分布と異なる Flash Crowds が発生した場合に満足な効果が発揮できない，3) ミラーサーバを設置する時期の判断が難しいといった問題がある．

これらの問題点を解決するには，サービスの需要変動をきめ細かく検知しミラーサーバの数や位置を需要に見合うように動的に変更する必要がある．またこのようなミラーサーバ群がサービスの提供基盤として存在することで，サービスの提供者は独自に Flash Crowds の対策を施す必要がなくなる．

本論文では，Flash Crowds に対応した動的再配置可能なミラーサーバ基盤として，ExaPeer を提案する．ExaPeer 上では，サービス提供者が提供するサービスの需要の規模や分布を事前に把握する必要はない．ExaPeer ではあらかじめインターネット上に多数の物理マシンを配備しておき，その物理マシン上でクライアント向けのサービスを提供する仮想マシンを実行する．この仮想マシンが，従来でいうサービスを提供するミラーサーバに相当する．ExaPeer はサービスの需要変動に応じて，サービスを提供する仮想マシンの数や位置を動的に変更する．また，あらかじめ配置した物理マシン群を有効に活用するために，ExaPeer を複数のサービスが利用可能な基

盤と位置付ける．需要分布の異なるサービスを同時に ExaPeer 上で展開することで，有休状態にある物理マシンをできるだけ減らすことができる．

適切な物理マシン上でミラーサーバを起動するために，ExaPeer では配置した物理マシン間でオーバレイネットワークを構築する．オーバレイネットワーク上でクライアントのアクセスを中継することで，アクセス経路を特定しこれが集中する点にミラーサーバを起動する．アクセス経路の特定のために，物理ネットワーク上で RTT が小さい物理マシンが，オーバレイネットワーク上でより近接することを保証する．このオーバレイネットワークの構築に，既存技術である Content-Addressable Network (CAN) [2] および Global Network Positioning (GNP) [3] を用いる．CAN は分散ハッシュ検索 [4] の一手法であり，検索クエリのルーティングテーブルを構築するために，構成する物理マシンに座標を与え論理的な位置関係を定義する．GNP はある仮想的な座標空間を構築し，物理マシン間の RTT をユークリッド距離で近似する．ExaPeer では GNP で生成した座標系を用いて CAN を構築する．

シミュレーションを行うことにより ExaPeer を評価した．行ったシミュレーションの条件下では，ExaPeer が，サービスを受けるまでにかかる遅延時間の平均値が元のサーバの位置より平均 82 %減少する位置に，ミラーサーバを配置することを確認した．

2 基礎技術

ExaPeer ではミラーサーバの配置や探索のための構造に CAN を，物理マシンやクライアントの位置関係の近似に 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 が既知の場合は本章は読み飛ばして構わない．

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，各物理マシン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物理マシン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 の物理マシンは，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.zip 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 に対応する物理マシンに送信されたとすると，ゾーン C に対応する物理マシン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 に対応する物理マシンは隣接するゾーン A が (2.5, 0.5) を含むので，ゾーン A に対応する物理マシンにクエリを転送する．

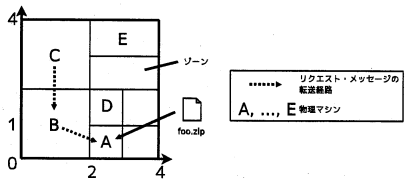


図 1: Content-Addressable Network

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，

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

を最小化するように座標を定める．ここで $d_{L_i L_j}$ は L_i, L_j 間の RTT，$\hat{d}_{L_i L_j}$ 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 [5] 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

発生の予測が難しい Flash Crowds に対応するには，その発生の検出を迅速に行えなければならない．

需要変動の迅速な検出のためには，需要変動の監視をできるだけ低負荷かつ頻繁に行える必要がある．需要変動を把握する一つの方法は，コンテンツの提供元であるサーバやそのミラーサーバのアクセス記録を収集し，これを分析することである．しかしこの方法は，アクセス記録の分析が高負荷になってしまう．特にサービス数の増加に対して負荷は線形かそれ以上の計算量で増加し，高頻度での需要変動の監視が難しくなる．ExaPeer では一極に集中しての監視ではなく，ExaPeer を構成する個々の物理マシンが独立して周辺の需要変動を検出する方式を採る．この方式では，検出対象とする範囲を物理マシンの周辺程度に限定することができ，需要変動を監視する負荷は比較的低い．

本章では，物理マシンが個別に需要変動を検出できるようにした ExaPeer の基本設計と，個々の物理マシンが行う需要変動検出の手法について述べる．

3.1 基本設計

需要変動を把握する指標として，ExaPeer はクライアントのアクセス経路の集中度を採る．ここでいうアクセス経路とは，クライアントがサーバにアクセスするまでに辿ったネットワーク経路のことであり，アクセス経路の集中度とは，経路上のある点において同じアクセス経路を辿ったクライアントの数を指す．この指標であれば，値の高低がミラーサーバを設置する効果の大きさに直結するので，ミラーサーバ設置の判定処理を単純化できる．

ExaPeer では，実際のネットワークにおける物理マシンの位置関係を反映したオーバレイネットワークの構築に，GNP で生成した座標で構築した CAN を用いる．GNP で生成した座標を用いることで，CAN 上の物理マシンの位置関係に実際のネットワーク上の位置関係を近似する．また，CAN 上でクライアントのサービスリクエストの転送を行うことで，個々の物理マシンによるアクセス経路の集中度の算出を可能にする．

また，ExaPeer では仮想マシン技術 [6] を用いてミラーサーバを物理マシンから独立することを想定する．サービスの提供部分を仮想マシンとして実現し，ハードウェアと分離する．

図 2 に，ExaPeer においてクライアントの送信したサービスリクエストが転送される様子を示す．まず，ExaPeer では提供するサービスごとに親となる物理マシンを決定する．親となる物理マシンの決定は ExaPeer がオーバレイネットワークの構築に利用した CAN のプロトコルに基づく．図 2 では左下の物理マシンが親である．クライアントはサービスが必要になると，自分の属するゾーンの物理マシンにサービスのリクエストを送信する．サービスのリクエストを受信した物理マシンは，CAN のプロトコルに従って親である物理マシンを探索する．経路の途中にサービスを提供する物理マシンがあった場合はそこで探索を終了する．サービスを提供する物理

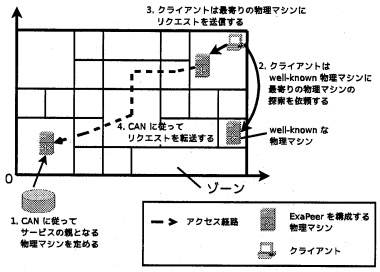


図 2: サービスリクエスト転送の様子

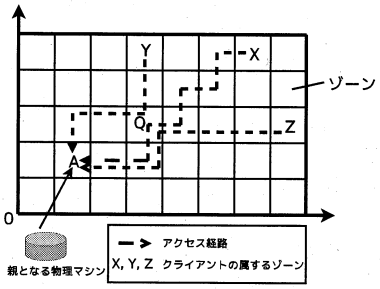


図 3: アクセス経路による需要分布の把握

マシンの IP アドレスをクライアントに送信し，クライアントはこの IP アドレスの指す物理マシンからサービスを受ける．なお，図 2 は 2 次元の CAN を用いた例である．本論文では ExaPeer の図示を 2 次元座標空間を用いて行う．

3.2 アクセス経路を用いた需要分布の把握

ExaPeer ではアクセス経路から需要分布を把握する手法を採用する．この手法の概要は次の通りである．物理マシンはサービス毎に，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，その地域からのアクセス経路上にある物理マシン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ておいた閾値を超えると，物理マシンはサービスを提供すると効果が大きいと判断する．

図 3 にアクセス経路から需要分布を把握する例を示す．破線の矢印はアクセス経路を示す．この例において，アクセス経路の集中しているゾーン Q の物理マシンの転送数は 3 であり，転送経路の集中していない他のゾーンの物理マシンは 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．

3.2.1 需要分布を詳細に把握する工夫

以上に示した方式で，狭い地域に局所的に発生する急激で大規模な需要変動には対処できる．しかし広範囲に起こる小規模な需要変動では，この素朴な方式では需要分布の把握を失敗することがある．

図 4 は需要分布の把握に失敗する例である．図 3 とは異なり，3 つのアクセス経路すべてが集中しているゾーンがない．図 3 と同じ需要分布であるにも

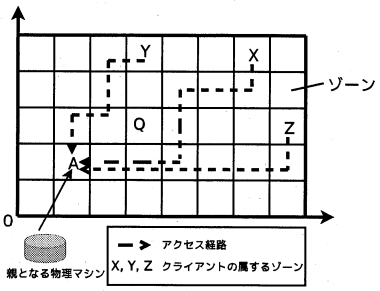


図 4: 需要分布の把握が失敗する例

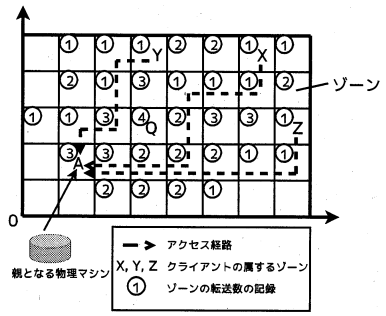


図 5: アクセス経路に隣接するゾーンに転送数を加算する

かかわらず，どの物理マシン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二つある．一つは，需要の高い地域が親となる物理マシンのあるゾーンから遠かったり，座標空間が高次元であるために，アクセス経路に揺らぎが生じてしまうためである．もう一つは，そもそも需要のある地域が散らばっていてアクセス経路の集中する点がないためである．

アクセス経路を絞りこむために，例えば転送する候補となるゾーンがいくつかある場合 x 軸方向から近づけるといった，転送の条件の範囲を絞りこむ方法がある．しかしこの方法は，負荷分散や可用性の面から採用できない．特定の物理マシンにアクセス経路が集中し負荷が偏ってしまったり，経路の途中にある物理マシンが異常停止したときに転送が中断してしまい，サービスが提供できなくなるおそれがあるからである．

アクセス経路の揺らぎによる弊害を低減するために，ExaPeer ではアクセス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 4 においてゾーン X からのアクセス経路はゾーン Q を含んでいない．ここでアクセス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 がアクセス経路に含まれているとみなすことができる．このようにしてアクセス経路に広がりを持たせ，経路の重なりが現れやすいようにしている．なお，アクセス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 5 は，図 4 で経路に隣接する物理マシン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 の物理マシンは図 3 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．

需要分布の把握が失敗する二つ目の原因は，先にも述べたとおり，需要のある地域が散らばっていてアクセス経路の集中する点がない場合である．このような場合でもうまく需要のある地域に近づけるようにミラーサーバを配置する必要がある．そのためには需要のある地域の位置を，物理マシンがなんらかの方法で知らなければならない．

ExaPeer ではリクエストの転送数のみを記録するという条件を保ちつつ，需要のある地域の方向に位置する物理マシンにミラーサーバが配置されやすくなるようにしている．具体的には，ミラーサーバが稼働中である物理マシンへリクエストを転送したときは，加算する転送数を 2 倍することで実現している．例えば図 4 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．CAN のプロトコルから，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，ミラーサーバが稼働中であるゾーン A へ転送した物理マシンが記録する転送数を 2 倍することでミラーサーバが配置されやすくなる．このようにすることで，ミラーサーバが稼働中の物理マシン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 6 は，図 5 の例でミラーサーバが稼働中の物理マシンへ転送した際は処理数を 2 倍とした場合を示す．ミラーサーバが稼働中であるゾーン A の周辺にあるゾーンの物理マシンのうち，アクセス経路の起点であるゾーン X, Y, Z の方向にあるゾーンの物理マシン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物理マシンの転送数は 2, 2, 0, 0, 0 である．よって，アクセス経路の起点に近い物理マシンの転送数の記録が大きくなっていることがわかる．

3.3 ミラーサーバの動的再配置

ExaPeer は 3.2 節で述べた転送経路による需要変動の把握手法に基づいて，ミラーサーバを動的に再配置する．ここではその方法について述べる．

物理マシンは 3.2 節で述べた方式に沿って，ミラーサーバ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，あらかじめ配置して

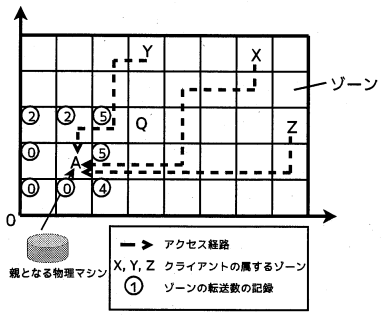


図 6: ミラーサーバが稼働中の物理マシンへ転送した際は 2 倍とする

表 1: 使用したネットワークトポロジ

Transit domain 数	228
1 domain あたりの Transit ノード数	5
1 Transit ノードあたりの Stub domain 数	4
1 Stub domain あたりのノード数	2
総ノード数	9,140
GNP 座標が異なるノード数	3,081

おいた仮想マシンのイメージファイルを元に仮想マシンを起動し，ミラーサーバとしての動作を開始する．転送数があらかじめ設定しておいた閾値 *LOWER* を下回ると，物理マシンは仮想マシンを停止し，ミラーサーバとして振る舞うことを中止する．

4 実験

ネットワークシミュレータ上に ExaPeer を実装し，シミュレータ上で ExaPeer に Flash Crowds を想定した負荷を与えた．評価は，クライアントとミラーサーバ間の遅延時間，同時に起動しているミラーサーバの数の点から行った．

ExaPeer の評価は，オーバレイネットワークの評価用のシミュレータである Overlay Weaver [7] を用いて行った．評価用のネットワークトポロジには Transit-Stub モデル [8] を用いた．Transit-Stub モデルとは，インターネットのような大規模ネットワークを階層的なツリー構造としてモデル化したものである．表 1 は，使用した Transit-Stub モデルを示したものである．表 1 が示す Transit-Stub モデルは，比較的大規模なバックボーンを備え，末端のネットワークが比較的小規模であるようなトポロジを想定している [9]．用意した Transit-Stub モデルの全ノードに 6 次元の GNP 座標を生成した．この中から同一の GNP 座標をもつノードを除いた 3,081 ノードを選択し実験に使用した．

4.1 実験 A: 広範囲からのアクセス

全ネットワーク規模で需要が変動したときの ExaPeer の挙動を調べるために，次のような設定でシミュレーションを行った．表 1 のネットワークトポロジから末端のノードを選択し，これをクライアントのアクセスを受けた ExaPeer の物理マシンと見なす．具体的には，他のノードとの接続が 1 つしかないノードを末端として選択した．実験の条件および ExaPeer のパラメータは表 2 の通りとした．WND_SIZE は記録ウィンドウのサイズ，CHK_INT は需要のチェック間隔を示す．また，比較のためにミラーサーバを起動しない場合でのシミュレーションも行った．

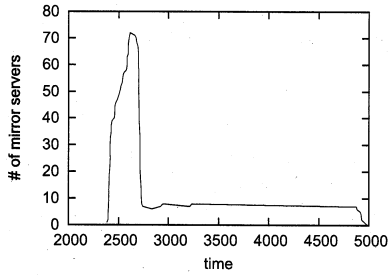


図 7: ミラーサーバ数の推移（実験 A）

図 7 は，ExaPeer 上で稼働中のミラーサーバ数の推移を示す．クライアントのアクセスが始まるとミラーサーバの数は増加し，2,615 秒経過時に 72 台まで増加した．その後ミラーサーバ数は減少し，7 から 8 台程度で安定した．これは高需要の発生直後は大量のミラーサーバを配置し，余剰分のミラーサーバを順次停止していくという ExaPeer の動作を示している．

図 8 は，クライアントがサービスを受けるまでの遅延時間の推移を示したものである．a) はミラーサーバを配置しない場合，b) は ExaPeer でミラーサーバを動的に再配置した場合である．2,500 秒経過時点前後から，遅延時間の平均値や分散が a) と比較して b) は減少していることがわかる．このことから，ミラーサーバが需要の高い地域の付近で起動し遅延時間が減少したと考えられる．b) は a) と比較して遅延時間の平均値は約 75 %減少し，分散は約 96 %減少した．

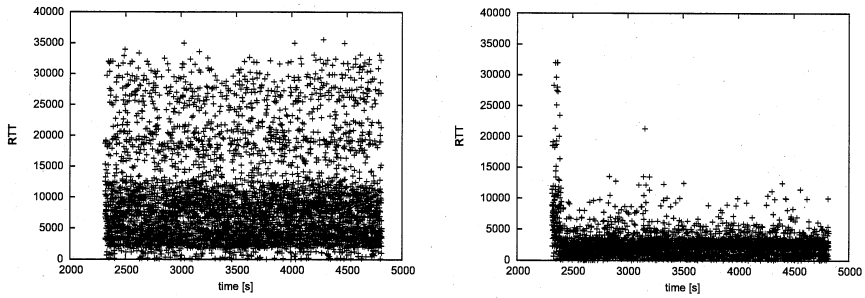
4.2 実験 B: 局所的なアクセス

小規模の需要が発生し，その分布が変動する場合における ExaPeer の挙動を調べるため，以下のシミュレーションを行った．実験 A で用いたノード群から 10 個のクラスタを構成した．クラスタの構成は，互いの RTT が 5,000 ミリ秒を超えるようなノードを中心とし，そこから RTT が 2,000 ミリ秒以内であるノードを 10 個選択することで行った．

シミュレーション開始後，10 個のクラスタを順次選択し，クラスタごとにランダムに選択したノードからクライアントのリクエストを送信した．リクエストを送信するクラスタの切り替えは 5,000 秒ごとに行った．実験の条件および ExaPeer のパラメータは表 2 の通りとした．また，比較のためにミラーサーバを起動しない場合でのシミュレーションも行った．

表 2: 実験の条件と ExaPeer のパラメータ

	負荷開始	負荷終了	REQ 間隔	UPPER	LOWER	WND_SIZE	CHK_INT
実験 A	2,145 sec	4,644.5 sec	500 msec	300	150	60 sec	300 sec
実験 B	2,366 sec	52,366 sec	500 msec	300	200	60 sec	300 sec
実験 C	2,145 sec	4,644.5 sec	500 msec	300	150	60 sec	300 sec
実験 D	2,145 sec	4,644.5 sec	500 msec	300	150	60 sec	300 sec



a) ミラーサーバなし　　b) ExaPeer

図 8: サービスを受けるまでの遅延時間の推移（実験 A）

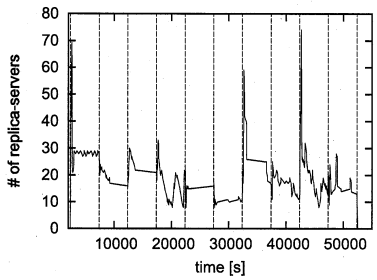


図 9: ミラーサーバ数の推移（実験 B）

図 9 は，ExaPeer 上で稼働中のミラーサーバ数の推移を示す．縦の破線はクラスタが切り替わった時点を示す．要求メッセージを送信するクラスタが切り替わると，それに追随してミラーサーバの数が増減している様子が読み取れる．また，多くのクラスタでミラーサーバの数が安定したこともわかる．

図 10 は，クライアントがサービスを受けるまでの遅延時間の推移を示したものである．a) はミラーサーバを配置しない場合，b) は ExaPeer でミラーサーバを動的に再配置した場合である．a) と比較して b) の分布の範囲が小さいことがわかる．また a) は最低でも約 2,000 ミリ秒の遅延時間が発生していることがわかる．b) は a) と比較して遅延時間の平均が約 82 %，分散が約 98 %減少した．

4.3 実験 C: 固定配置方式との比較

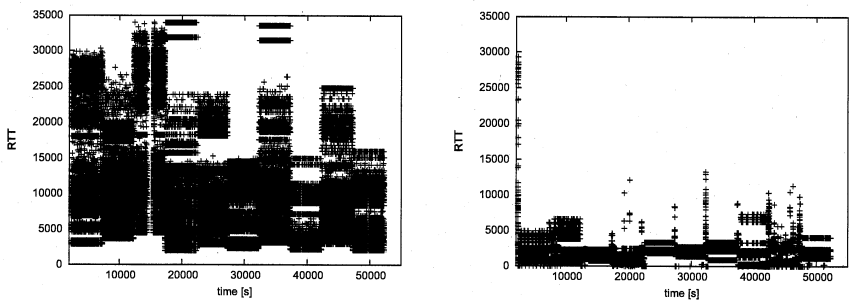
ExaPeer とミラーサーバを固定配置した場合との比較を行った．実験の条件および ExaPeer のパラメータは表 2 の通りとした．固定配置するミラーサーバの数は，同条件下で ExaPeer が配置した数が 7〜8 台であったので，それを若干上回る 10 台とした．固定配置の方法は，1) ランダムに選択する方法，2) ネットワークトポロジ上の接続ノード数が上位のノードから順に選択する方法の 2 つを採用した．

図 11 はそれぞれの方式についてクライアントがサービスを受けるまでの遅延時間の推移を示す．a) は ExaPeer，b) はランダムに選択する方法，c) はネットワークトポロジ上の接続ノード数が上位のノードから順に選択する方法である．b) や c) に比べて a) は分布が低い値に集中していることがわかる．a) は b) および c) と比較して，遅延時間の平均は約 75 %，分散は約 96 %低かった．

4.4 実験 D: 需要変動把握機能の効果検証

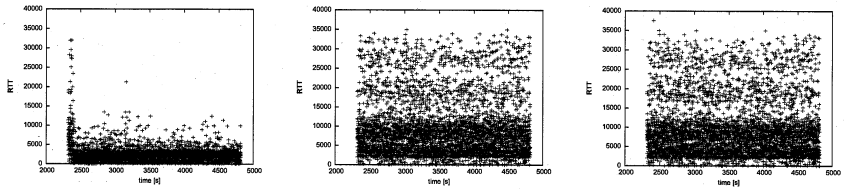
ExaPeer に採用した機能の効果を確かめるため，各機能を有効にした場合と無効にした場合との比較を行った．ここでいう ExaPeer の機能とは，CAN の座標系として採用した GNP と，3.2.1 節で示した 2 つの工夫である．実験の条件および ExaPeer のパラメータは表 2 の通りとした．

図 12 は，クライアントがサービスを受けるまでに要した遅延時間を，a)ALL-ON: すべて有効にした場合，b)T-OFF: サービスを提供中の物理マシンの直前にあたる物理マシンで転送数を 2 倍する機能を無効にした場合，c)T-N-OFF: b) に加えて隣接ノードに転送数を加算する機能を無効にした場合，d)ALL-OFF: c) に加えて CAN の座標系に GNP を使用せずランダムとした場合とで比較したものである．c) や d) は，a) に比べて遅延時間の分散は終止高いままである．これは需要の発生している地域へのミラーサーバの配置が行えていないことを示している．b) は c) や d) に比べると分散は小さいので，



a) ミラーサーバなし　　b) ExaPeer

図 10: サービスを受けるまでの遅延時間の推移（実験 B）



a) ExaPeer　　b) ランダム　　c) 接続ノード数上位

図 11: サービスを受けるまでの遅延時間の比較（実験 C）

ある程度ミラーサーバを需要の発生地域に配置できているとみなすことができる．しかし，a) と比較して部分的に遅延時間が上昇している時がある．これは配置したミラーサーバが需要が低いと誤って判断して停止してしまい，ミラーサーバの数が安定しないからである．

5　関連研究

主に静的に生成されるコンテンツを効率よく配信するための機構として，Content Distribution Network (CDN) がある．CDN は主に Web サーバを支援するためのもので，画像や動画といった静的に生成される大規模コンテンツを対象にしている．インターネット上に複数の代理サーバを設置し，クライアントは最寄りの代理サーバにアクセスする．代表的な CDN として，商用サービスの Akamai [10] や，Globule [11]，CoralCDN [12] などがある．

Akamai では独自 DNS サーバを用いることで，クライアントに最寄りのキャッシュサーバを介してコンテンツの提供元にアクセスさせる．Akamai の方式では，Flash Crowds に対して十分な数のキャッシュサーバをあらかじめ用意しておく必要がある．ExaPeer は例え Flash Crowds の発生地域の物理マシン数が不足していても，その近傍にある物理マシンでミラーサーバを稼働することができるため，より Flash Crowds への耐性が高いといえる．

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．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 Crowds 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 Crowds に耐えられるように十分小さくする必要がある．

CoralCDN で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，クライアントに提供する．CoralCDN では設定した閾値以下の RTT であるノードから，優先的にコンテンツのキャッシュを探索する．そのためクライアントの密度は低いが総数は多いような Flash Crowds が発生すると，適切なキャッシュ配置がなされないおそれがある．ExaPeer はこのような Flash Crowds でも適切なミラーサーバの配置が期待できる．

CDN はコンテンツを高需要地域へ再配置することが主要な機能であるが，ExaPeer ではコンテンツを提供するサーバシステムそのものを再配置する．このことから，ExaPeer は次の二点において CDN と異なる．一点目はサーバの負荷分散を対象とする点である．CDN では動的に生成されるコンテンツに対しての対応がいまだ不十分であるため，コンテンツ生成に関する負荷分散が満足に行えていない．ExaPeer はコンテンツの生成に必要な機構を含めて分散配置するので，コンテンツの生成にかかる負荷の分散も期待できる．二点目は提供するコンテンツに関してユーザの自由度が大きいことである．最近の CDN では動的生成コンテンツについても対応が進んでいる．しかし，これらは CDN 側で用意した環境の上で動作することを前提にしているため，ユーザであるコンテンツの提供者の自由度は低い．ExaPeer は動的生成コンテンツに必要な生成系も含めて分散配置の対象であるため，ユーザが用意した環境が利用できる．

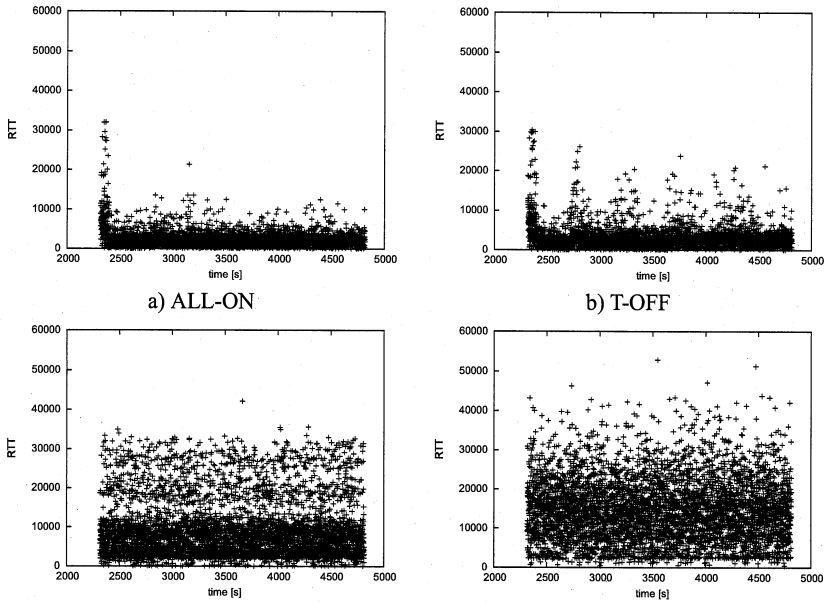


図 12: サービスを受けるまでの遅延時間の比較（実験 D）

6 おわりに

本論文では，Flash Crowds に対応した動的再配置可能なミラーサーバ基盤 ExaPeer を提案した．ExaPeer はサービスの需要変動に応じて，サービスを提供する仮想マシンの数や位置を動的に変更することで，擬似的にミラーサーバの移動や増減を実現する．

シミュレーションを行うことにより ExaPeer を評価した．シミュレーションの条件下において ExaPeer によるミラーサーバの配置は，需要の分布が変動しても遅延時間が平均 82 %低くなる位置にミラーサーバを動的に再配置することが可能であった．

今後の課題として，より詳細なシミュレーションを行う必要があると考える．例えば，サーバのアクセス記録からサービスの負荷情報を，ネットワークのトポロジ情報からホスト間の RTT 情報を取得し，これらの情報から ExaPeer に負荷を与え挙動を観測する．また，実際に発生した Flash Crowds のトラフィック情報から実験，分析を行えるとよい．

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] Jung, J., Krishnamurthy, B. and Rabinovich, M.: Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites., *Proc. of Int'l Conf. on World Wide Web*, pp. 293–304 (2002).

[2] Ratnasamy, S., Francis, P., Handley, M., Karp, R. M. and Shenker, S.: A Scalable Content-Addressable Network., *Proc. of SIGCOMM*, pp. 161–172 (2001).

[3] Eugene Ng, T. S. and Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches., *Proc. of IEEE Infocom*, pp. 170–179 (2002).

[4] 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).

[5] Szymaniak, M., Pierre, G. and van Steen, M.: Scalable Cooperative Latency Estimation., *Proc. of Int'l Conf. on Parallel and Distributed Systems*, IEEE Computer Society, pp. 367–376 (2004).

[6] Goldberg., R. P.: Survey of Virtual Machine Research, *IEEE Computer Magazine*, Vol. 7, No. 6, pp. 34–45 (1974).

[7] 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).

[8] Zegura, E. W., Calvert, K. L. and Bhattacharjee, S.: How to Model an Internetwork., *Proc. of IEEE Infocom*, pp. 594–602 (1996).

[9] Xu, Z., Tang, C. and Zhang, Z.: Building Topology-Aware Overlays Using Global Soft-Stat., *Proc. of Int'l Conf. on Distributed Computing Systems*, pp. 500–508 (2003).

[10] 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).

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

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