グループの概念を取り入れた P2P ウェブキャッシングシステムに向けた考察

岩丸晃大[†1] 糸川剛[†2]
北須賀輝明[†2] 有次正義[†2]

Peer to Peer(P2P) の技術を利用したものに P2P ウェブキャッシングシステムがある．従来の P2P ウェブキャッシングシステムでは特定の利用者に人気があるコンテンツでも，システム全体としては不人気のコンテンツの場合には，キャッシュに保持されずにミスヒットする可能性がある．また，各利用者の興味はそれぞれ異なっていることが一般的とはいえ，興味が似通っている利用者もいる．そこで本稿では，興味が似通った利用者でグループを構築する概念を取り入れた P2P ウェブキャシングシステムを提案する．従来のシステムと提案したシステムにおける性能を比較，評価するためにヒット率，ホップ数，メッセージ数を測定し，考察を行う．

A Consideration towards Constructing a P2P Web Caching System with Group Framework

Akihiro Iwamaru,[†1] Tsuyoshi Itokawa,[†2] Teruaki Kitasuka[†2] and Masayoshi Aritsugi[†2]

In recent years, P2P web caching systems have been proposed. The storage of the systems, however, may fail to keep holding such contents that are not so pupular to all the users but are accessed frequently by a part of them. Also, some users often have similar interests in the system. In this paper, we propose a P2P web caching system with group framework in order to adapt such situations. We compare our system with a conventional system in terms of the average hit rate, the number of hops, and the number of messages.

1. はじめに

現在，HTML ドキュメントなどのウェブのデータを一時的に記憶しておくウェブキャッシングの技術は幅広く活用されている．ウェブキャッシングを利用することによりネットワークのトラフィック，サーバの負荷を減らすことが可能であり，利用者が必要なデータを得るための時間も短縮される．

ウェブキャッシングにはクライアントマシン上のブラウザによるキャッシング，クライアントマシンとウェブサーバの間に位置するプロキシサーバによるキャッシングがある．プロキシサーバによるキャッシングのパフォーマンスは，プロキシサーバの利用者の数に依存し，利用者が増加する程，キャッシングされたデータが再びリクエストされる可能性が高まる．そのためにプロキシサーバによるキャッシングは，利用者が多く，ネットワークの通信速度が速い大学や会社等でよく利用される．

しかし，キャッシングに利用されるストレージには限りがあるので，プロキシサーバによるキャッシングのパフォーマンスにも限界がある．そのためプロキシサーバは，ストレージに新しいデータを保持するために古いデータを追い出す必要がある．しかし，ストレージが小さい場合，古いデータだけでなく，新しいデータも追い出す可能性が高くなるために，パフォーマンスを落とすことになる．

このストレージの問題点を改善するために，ウェブキャッシングをサポートした P2P(Peer to Peer) システムがある．この P2P ウェブキャッシングシステムでは，プロキシキャッシュシステムを利用したい利用者一人一人が自分のクライアントマシンのストレージをキャッシュシステムに提供する．この提供されたストレージを P2P システムによって利用者全員で共有し，要求されたデータを保持することに使用する．クライアントマシンはこのキャッシングシステムにおいて，どのクライアントマシンにデータが保持してある

†1 熊本大学工学部数理情報システム工学科
Department of Computer Science, Faculty of Engineering, Kumamoto University
†2 熊本大学大学院自然科学研究科
Graduate School of Science and Technology, Kumamoto University

のかを見つけるために，分散ハッシュテーブル (DHT) を利用する．このシステムはクライアントマシンが増えると，キャッシングに用いられるストレージも自動的に増える．そして，コストをかけることなく高いパフォーマンスを保つことが可能である．

利用者が増加することはシステム全体のストレージの増加につながるが，利用者は一人一人興味が異なっており，アクセスするオブジェクトは異なる．そのため，特定の利用者に人気があるコンテンツでも，システム全体としては不人気のコンテンツはストレージに保持されていない可能性がある．そこで，本稿では興味が似通った利用者でグループを構築することでこの問題を解決する手法の提案及び評価を行う．

本稿は次のような構成となっている．第 2 章において P2P ウェブキャシングに関して紹介する．第 3 章において本稿で紹介するグループを構築する提案手法を紹介する．第 4 章において従来手法と提案手法を比較した実験結果を評価する．第 5 章において結論と今後の課題を述べる．

2. 背　　景

P2P ウェブキャッシングシステムとは，システムに参加するクライアントマシンのストレージを P2P の技術を用いて共有することでプロキシキャッシュシステムを実現するシステムである．P2P ウェブキャッシングシステムは大きく 2 つに分類される．従来のプロキシサーバと P2P ウェブキャッシングシステムを組み合わせた Central server based system[1)2)] と，サーバを全く設置しないピュア P2P 型の Fully decentralized system[4)6)] である．

Central server based system では，クライアントは最初にプロキシサーバに対して欲しいオブジェクトを要求する．ここでミスヒットしてしまった場合に初めて，P2P ネットワーク上を探索する．また，ネットワークで接続されているクライアントのグループに対して接続しているプロキシサーバが全クライアントがキャッシュしているオブジェクトのインデックスファイルを管理するシステム[1)] であったり，プロキシサーバの負荷を分散させるためにスーパーノードを導入しインデックスファイルを分散して管理するシステムもある．また，プロキシサーバが DHT を用いてクライアントのストレージを探索するシステム[2)] も存在する．

Fully decentralized system には，Pastry[3)] のプロトコルを利用した Squirrel[4)] と呼ばれるシステムがある．また，Chord[5)] のプロトコルを利用した P2P ウェブキャシングシステム[6)] もある．

しかし，従来の P2P ウェブキャッシングシステムでは，特定のクライアントに人気があるコンテンツでも，システム全体としては不人気のコンテンツはストレージに保持されていない可能性がある．

3. 提 案 手 法

本研究では，グループを取り入れた P2P ウェブキャッシングシステムを提案する．利用者は一人一人興味が違っておりアクセスするオブジェクトが異なるが，利用者の中には興味が似通っている利用者もいる．この興味が似通っている利用者同士でグループを構築し，グループ用のストレージを確保する．このストレージにシステム全体としては不人気のコンテンツもグループ内において人気があれば，保持しておく．これよって，従来の P2P ウェブキャッシングシステムでは，ミスヒットしていた特定のクライアントに人気があるコンテンツが，ヒットすることになる．

本稿では，Chord のプロトコルを利用した Fully decentralized 型の P2P ウェブキャッシングシステムを元にグループを構築するものとする．

3.1 前　準　備

これから述べる提案手法の前準備として，Chord の説明を行う．Chord ではハッシュ値の計算に SHA-1 等のハッシュ関数が用いられる．ハッシュ関数に SHA-1 を用いる場合は，ハッシュ空間は 160 ビットのリング状の構造をとる．ハッシュ関数を用いて，ノードとデータの両方に対してハッシュ値を求める．Chord では，あるデータのハッシュ値が x であるとき，リング上の x の位置から時計回り (ハッシュ値が増加する方向) にたどっていった時に最初にあるノードがハッシュ値 x のデータを管理するノードとなる．図 1 において，黒い丸印がシステムに参加しているノードを表す．ここでは簡単に説明するためにハッシュ空間を 6 ビットと仮定して説明をする．ノードのハッシュ値が 21 であるノードを N21，オブジェクトのハッシュ値が 38 であるオブジェクトを K38 と表している．これ以降に示す図に関しても同様な表記法をとる．図 1 における実線の矢印は Chord においてオブジェクトを保持するノードを表しており，K10 は N14 に，K54 は N56 に保持される．

また，図 2 で示すように各ノードは自分のノードより 2^m $(m = 0, 1, 2 \cdots, 2^m)$ 個先のノードの情報の方を持たせる．これを Finger Table と呼ぶ．2^m 個先にノードが存在しない場合は，Chord リング上において時計回りの方向で隣接するノードの情報を持たせることにする．すると，Finger Table に含まれるノードにショートカットして探索することが可能になる．図 2 では，N8 は N14，N21，N32，N42 の情報を保持しておくことになり，探索の際には N14，N21，N32，N42 にショートカットすることができる．各ノードは $O(\log N)$ 個のノードの情報を保持する必要があるが，探索は $O(\log N)$ 回だけで済むため，非常に効率が良いと言える．

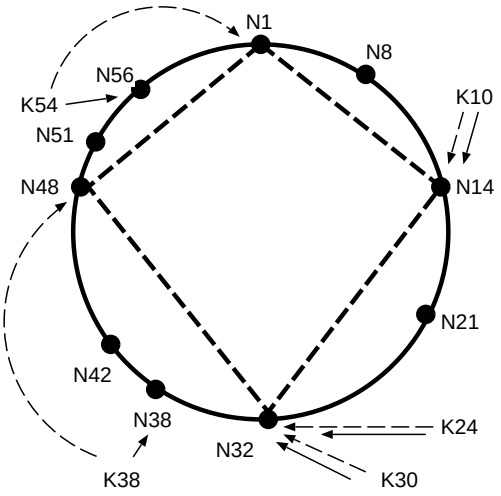

図 1 オブジェクトの保持場所

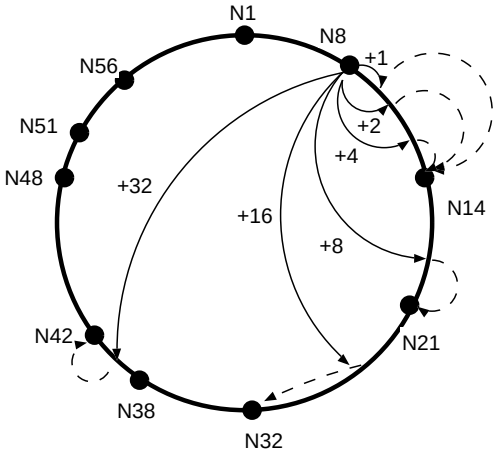

図 2 Chord の Finger Table

3.2 グルーピングポリシー

前述したように Chord はリング状の構造をしているが，グループを構築したノードはその Chord リングの内側にグループ用のリングを作り，マルチリング構造[7] にする．図 1，図 3 において，点線で結ばれているノードがグループを構築していることを表している．. 図 3 は N21 と N38 と N56，N1 と N14 と N32 と N48 で 2 つのグループを構築したときの構造である．グループ用に新たにハッシュ関数を用いてハッシュ空間を作成するのでは無く，Chord リングが作られているハッシュ空間内で新たなリングを作る．システムに参加しているノードが別のノードからグループへ参加要求を受け取ったら，グループを構築する．ただし，本研究では各ノードは高々一つのグループにしか所属できず，グループは少数のノードで構築されることを前提とする．

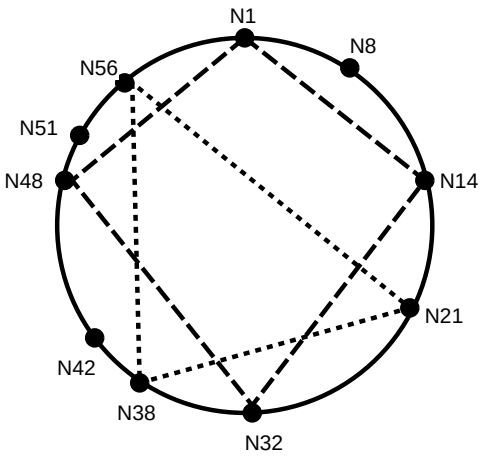

図 3 グループの構築例

3.3 ストレージポリシー

各ノードは Chord 用のストレージとグループ用のストレージの 2 つを準備する．ただし，グループに所属してないノードのストレージは，グループ用のストレージも Chord 用のストレージとして扱うことにする．もし，両方のストレージに同一のオブジェクトが保持される場合はグループ用のストレージには保持せずに，Chord 用のストレージだけに保持するものとする．また，グループ用のストレージに空きが存在する場合は，その空きの部分を Chord 用のストレージとして使用し，Chord 用のストレージに空きが存在する場合は，その空きの部分をグループ用のストレージとして使用することにする．

グループに所属するノードが要求したオブジェクトがキャッシュされておらずミスヒットした場合，そのオブジェクトはグループ用のストレージに保持しておく．グループ用のストレージにオブジェクトを保持するように要求があった場合には，オブジェクトのハッシュ値の位置からグループリング上を時計回りにたどっていった時に最初にあるノードがそのオブジェクト保持する．図 1 における点線矢印はグループ内でオブジェクトを保持するノードを表している．Chord では K38 は N38 に，K54 は N56 に保持されるが，グループでは K38 は N48 に，K54 は N1 に保持される．

3.4 リプレイスメントポリシー

リプレイスメントのアルゴリズムには，LRU 方式を利用する．各ノードは Chord 用のストレージとグループ用のストレージの 2 つのストレージを持つが，それぞれ LRU のリプレイスメントアルゴリズムに従うものとする．

グループ用のストレージに空きが存在し空きの部分を Chord 用のストレージとして使っていた場合に，グループ用のストレージにオブジェクトを保持するように要求があると，Chord 用のストレージとして使って

いる中でリプレイスメントを行う．また，Chord 用のストレージに空きが存在し空きの部分をグループ用のストレージとして使っていた場合に，Chord 用のストレージにオブジェクトを保持するように要求があると，グループ用のストレージとして使っている中でリプレイスメントを行う．

3.5 ルックアップポリシー

グループに所属しているノードは，グループリングを探索し，その後 Chord リングを探索する．グループリングの探索は，グループリング上を時計回り方向に線形探索するものとする．グループに所属していないノードは，Chord リング上の探索を行う．Chord リング上の探索は従来の Chord の探索方法を用いる．

グループに所属しているノードは，Finger Table だけでなく，グループリングにおいて隣接するノードの情報を保持しておく必要がある．そのノード情報は Chord リング上の探索の時には使わず，グループリングの探索の時にだけに用いるものとする．そのため，グループリングを使ったショートカットは行われない．

グループに所属しているノードがオブジェクトを得るために要求を出すと，まずグループリング上で欲しいオブジェクトを保持しているノードを探索する．ここで探していたオブジェクトが見つかりヒットすれば問題ないが，ミスヒットした場合はオブジェクトを要求したノードから Chord リング上の探索を開始する．

Chord リング上でヒットした場合はオブジェクトの要求を出したグループ内にオブジェクトを保持しておくように要求を出す．グループ内での探索がミスヒットし，Chord リングでの探索もミスヒットした場合は，グループ内と Chord に対してオブジェクトを保持するように要求を出す．

4. 実　　験

グループの概念を取り入れた提案手法の有効性を評価するため，オーバレイ構築ツールキット Overlay Weaver[8] を用いて提案手法のシュミレータを実装した．このシュミレータを用い，ヒット率，ホップ数，メッセージ数を測定し本来の Chord の手法と提案手法を比較する．

Zipf 分布[9] に基づいてどのオブジェクトがどれほどの頻度で要求されるのかを決定した．Zipf 分布は以下の式で表される．

$$f(k, \alpha, N) = \frac{k^{-\alpha}}{\sum_{n=1}^{N} n^{-\alpha}} \qquad (1)$$

本実験では，上式において $N = 2500000$，$\alpha = 1.0$ とし，700000 回のアクセスを発生させる．

各グループは 5 ノードで構築されるものとし，各グループのアクセスに偏りを発生させる．システム全体のノード数に対する各グループに参加しているノード数の割合と，全てのオブジェクト数に対するグループ内で人気のあるオブジェクト数の割合を一致させる．例えば，システム全体のノード数が 100 の時にグループを 2 つ構築した場合には，1 つのグループではシステム全体として 1，21，41，61，81，101，…番目に人気のあるオブジェクトをそのグループ内で人気のあるオブジェクトとし，もう 1 つのグループでは．システム全体として 2，22，42，62，82，102，…番目に人気のあるオブジェクトをそのグループ内で人気のあるオブジェクトとする．

そして，グループ内で人気のあるオブジェクトはグループ内のノードからアクセスをされるようにする．グループ内で人気のあるオブジェクトに対するアクセスが全てグループ内のノードだけからアクセスされるのではなく，グループ外のノードからもアクセスされることがあるのが一般的である．そこで，グループ内で人気のあるオブジェクトは 2 回に 1 回の割合でそのグループ内のノードからアクセスされるようにした．

本実験では，各ノードのストレージのキャパシティは管理するオブジェクトの総サイズではなく，オブジェクトの総数で表現する．また，最大 50 個のオブジェクトを保持できるものとする．各ノードの初期状況はストレージには何も保持されていないものとする．本実験による測定は，グループを構築し終えてから開始し，グループ構築後はノードの参加・離脱は考えず，ノード数は変化しないものとする．

システム全体のノード数，グループ用のストレージのキャパシティ，各グループを構築しているノード数，システム全体のノード数に対するグループに所属しているノード数の割合を変化させ測定を行った．

4.1 ヒット　率

システムに参加しているノード数を 100，200，300，400，500 と変化させた時のシステム全体のヒット率，グループに所属するノードのヒット率，グループに所属していないノードのヒット率及びグループを構築しない場合のヒット率を比較し，提案手法がヒット率に与える影響を調査する．

ここで，グループに所属しているノード数の全体に対する割合を G_r，グループのキャパシティを C_g と表す．

G_r を 25%，50%，75%，100%とした場合を測定した結果を，図 4 から図 7 に示す．

図 4 と図 5 から本来の Chord の手法と提案手法のシステム全体のヒット率は，ほとんど変わらず，グループに所属しているノードのヒット率が従来よりも向上していることがわかる．しかし，図 6 と図 7 では，システム全体のヒット率だけでなく，グループに所属しているノードのヒット率も低下している．これはグループ用のストレージを作ることによって，グループリング上の探索時にグループ内において人気の高いオブジェクトにヒットすることが増えるが，Chord リン

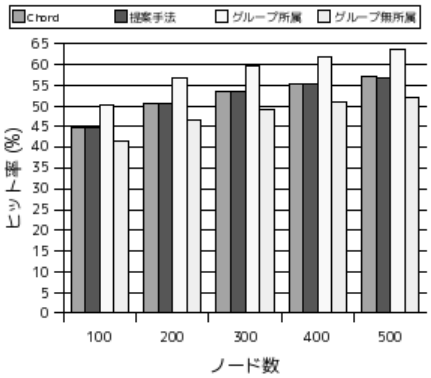

図 4 $G_r = 25\%$, $C_g = 5$ の場合のヒット率

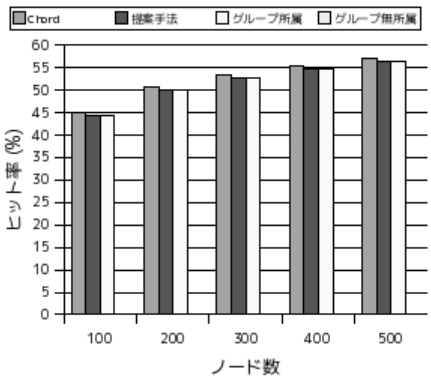

図 7 $G_r = 100\%$, $C_g = 5$ の場合のヒット率

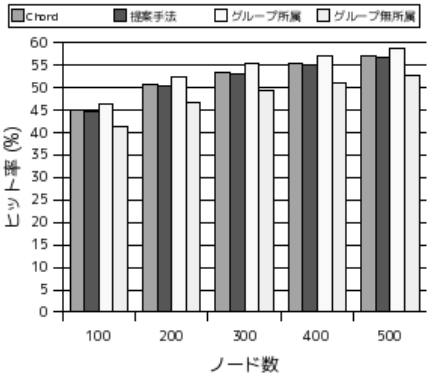

図 5 $G_r = 50\%$, $C_g = 5$ の場合のヒット率

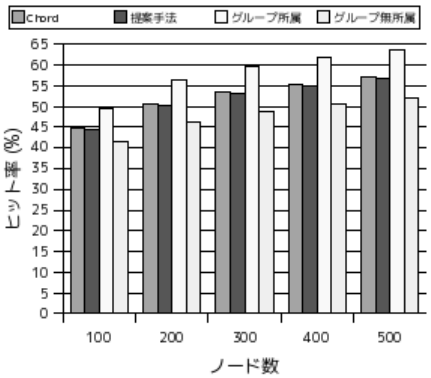

図 8 $G_r = 25\%$, $C_g = 10$ の場合のヒット率

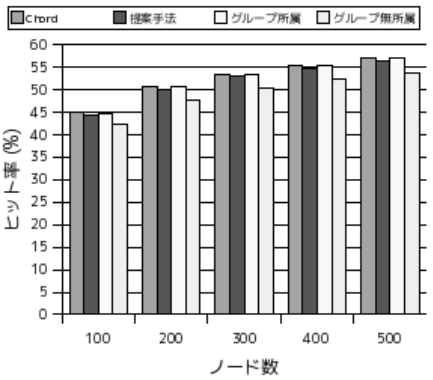

図 6 $G_r = 75\%$, $C_g = 5$ の場合のヒット率

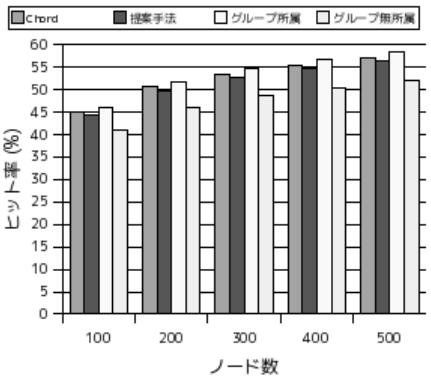

図 9 $G_r = 50\%$, $C_g = 10$ の場合のヒット率

グ上の探索の時にはミスヒットすることが増えることを意味している．また，システム全体としてヒット率が向上していないのは，Chord 用のストレージに保持されているオブジェクトがグループ用のストレージにも保持されることで，結果としてストレージを無駄に使っていることが原因として考えられる．

次に，グループのキャパシティを変化させた時のヒット率を比較するため，G_r が 25%，50%の場合において，C_g が 10，15 の場合を測定した．その結果を図 8 から図 11 に示す．

図 4 と図 8 と図 10，図 5 と図 9 と図 11 をそれぞれ比較してみると，グループ内のキャパシティが増え

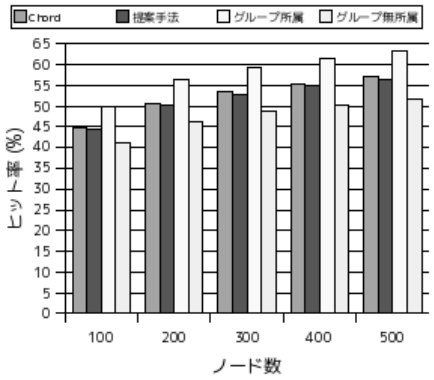

図 10 $G_r = 25\%$，$C_g = 15$ の場合のヒット率

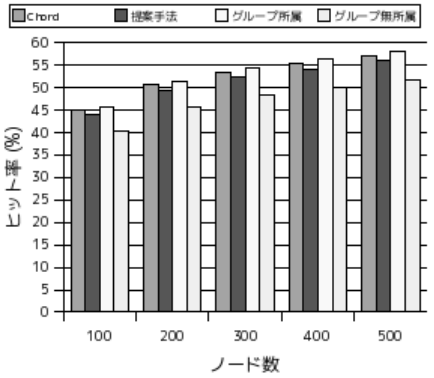

図 11 $G_r = 50\%$，$C_g = 15$ の場合のヒット率

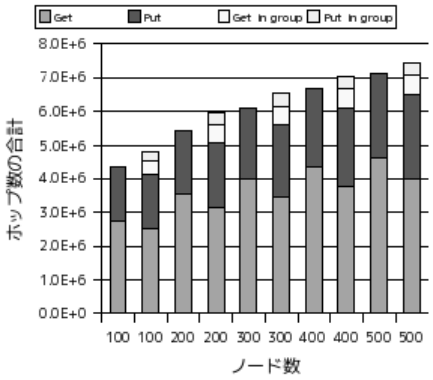

図 12 $G_r = 25\%$，$C_g = 5$ の場合のホップ数

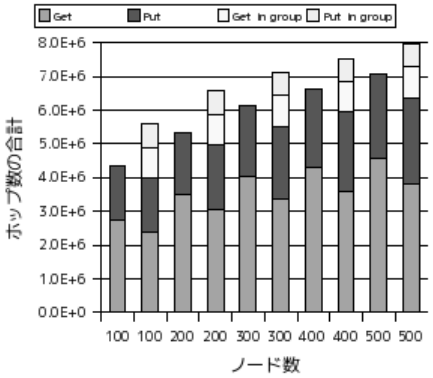

図 13 $G_r = 50\%$，$C_g = 5$ の場合のホップ数

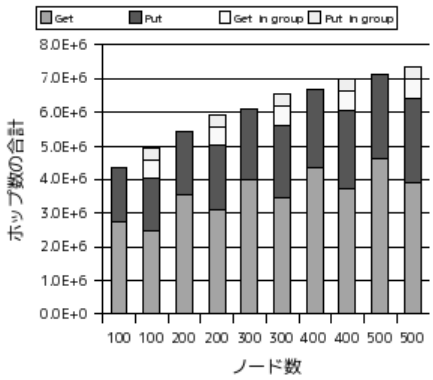

図 14 $G_r = 25\%$，$C_g = 10$ の場合のホップ数

てもヒット率が向上せずに低下していることがわかる．これは人気のあるオブジェクトはグループ用のストレージが少なくても保持されているためである．また，グループ用のストレージを大きくすると，Chord 用のストレージは減り Chord リング上を探索する時にミスヒットすることが多くなる上，グループ用のストレージには比較的人気がないオブジェクトも保持されることが多くなるからである．

4.2 ホップ 数

オブジェクトが保持されているノードを探索 (Get) する時のホップ数，オブジェクトをどのノードに保持するのかを探索 (Put) する時のホップ数，グループ内において Get，Put する時のホップ数を測定し，提案手法がホップ数に与える影響を調査する．システムに参加しているノード数を 100，200，300，400，500 と変化させた時のホップ数を提案手法と Chord で比較する．

G_r が 25%，50%の場合に，C_g が 5，10，15 の場合を測定する．

測定した結果を図 12 から図 17 に示す．ホップ数はグループを構築した提案手法の方が多くなった．これは，グループ内での探索が線形探索であることが大きな要因である．図 12 と図 13，図 14 と図 15，図 16 と図 17 を比較すると，グループに所属しているノード数が 2 倍になったためグループにおける Put，Get の際のホップ数が 2 倍近くになっている．これは単純にグループに所属しているノード数が 2 倍となり，探

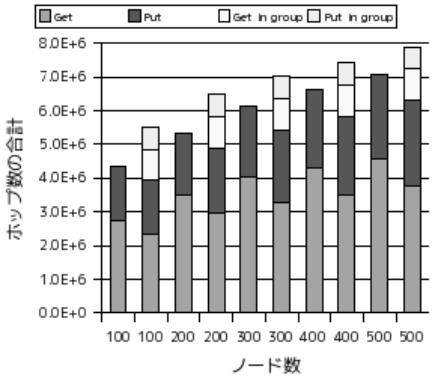

図 15 $G_r = 50\%$, $C_g = 10$ の場合のホップ数

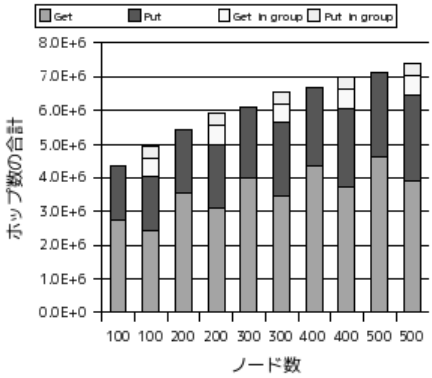

図 16 $G_r = 25\%$, $C_g = 15$ の場合のホップ数

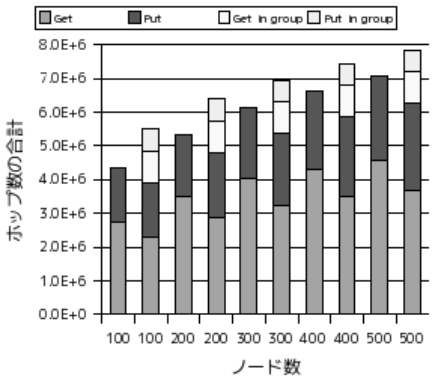

図 17 $G_r = 50\%$, $C_g = 15$ の場合のホップ数

索する機会が 2 倍となったからである．

いずれの場合もグループ内でヒットすることにより Chord 上を探索することが減るため，グループを構築しない場合よりも Get の際のホップ数が減っている．一方，Put の際のホップ数はほとんど変化がない．これは前述した通りシステム全体のヒット率がグループを構築したときもほぼ変わらず，ミスヒットした時だけにしか Put が行われないからである．

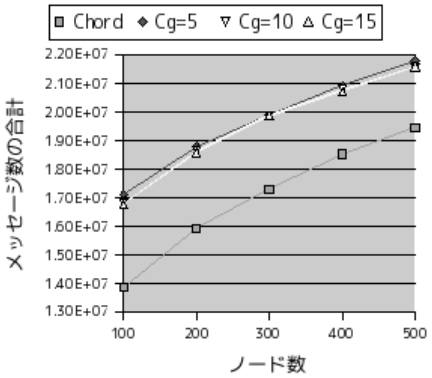

図 18 $G_r = 25\%$ の場合のメッセージ数

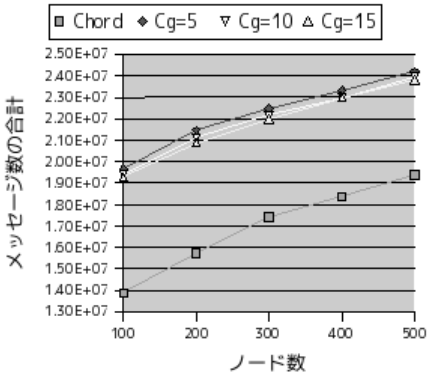

図 19 $G_r = 50\%$ の場合のメッセージ数

図 12 と図 14 と図 16，図 13 と図 15 と図 17，を比較すると，グループのキャパシティが増えるとホップ数の合計が若干減少しているのがわかる．これはグループのキャパシティが大きいとグループ内の探索でヒットする可能性が高まるからである．前述した通りグループのキャパシティを大きくすることで Chord 用のストレージが減り，ヒット率が低下するが，グループ内での探索においてのヒット率が増加しホップ数は減少するという利点がある．

4.3 メッセージ数

ネットワークのトラフィックを表す指標としてメッセージ数を測定した．グループを構築しない本来の Chord と C_g が 5，10，15 の場合をノード数を変化させて比較した．さらに，G_r が 25%，50%の場合を測定した．この実験により，提案手法がメッセージ数に与える影響を調査する．

測定結果を図 18 と図 19 に示す．

いずれの場合も，提案手法の方がメッセージ数は多くなる．グループのキャパシティの変化によってメッ

セージ数はそれほど変わらない．Chord と提案手法を比べると図 18 ではメッセージ数におよそ 2.5×10^6 の差がある．同様に比較すると図 19 ではメッセージ数がおよそ 5.0×10^6 これは前述したグループ内におけるホップ数の場合の関係と同じで，各 G_r が 50%の場合は，G_r が 25%のおよそ 2 倍になっている．さらに，グループのキャパシティによってメッセージ数はそれほど変わらないが，グループのキャパシティが大きい方がメッセージ数が少ない．このことも，グループ内におけるホップ数に関係があるといえる．つまり，提案手法とのメッセージ数の違いはグループ内の探索手法が原因である．グループ内の探索手法を改良することにより，ホップ数とメッセージ数を共に減らすことができると期待できる．

5. おわりに

P2P ウェブキャシングシステムにおいて，特定の利用者に人気があるオブジェクトでも，システム全体としては不人気なオブジェクトはストレージに保持されていない問題を解決するために，興味が似通った利用者でグループを構築する手法を提案した．

実験からグループを構築した場合のヒット率，ホップ数，メッセージ数の変化を調査し，少数のノードがグループに所属している場合にはシステム全体のヒット率をほとんど下げることなくグループに所属しているノードのヒット率が向上することがわかった．

しかし，問題点として，グループに所属してないノードのヒット率が低下することが挙げられる．この問題点は，グループに所属していない利用者に不公平感を感じさせないようにするために，他のノードよりもシステムに貢献しているノードがグループに所属できることにするなどの工夫を検討する余地があることを示唆している．また，他の問題点として，多数のノードがグループに所属した場合にはグループに所属していてもヒット率が低下することが挙げられる．また，ホップ数やメッセージ数が増加することの 2 点が挙げられる．ヒット率低下の原因は，Chord 用のストレージに保持されているオブジェクトをグループ用のストレージにも保持することによるストレージの無駄遣いである．また，ホップ数とメッセージ数の増加はグループ内の探索手法が原因であると考えられる．

今後の課題として効率の良いグループ内の探索手法を採用し，Chord 用のストレージに保持されているオブジェクトはグループ用のストレージに保持せずに，グループ用のストレージにはグループ内のノードだけに必要されるオブジェクトを保持するようなリプレイスメントポリシー，ルックアップポリシーに改良することが挙げられる．

謝辞 本研究の一部は，科学研究費補助金基盤研究 (C)(18500073) により行われた．

参 考 文 献

1) L.Xiao, X. Zhang, and Z. Xu, “On Reliable and Scalable Peer-to-Peer Web Document Sharing”, *Proc. 16th International Parallel and Distributed Processing Symposium (IPDPS’02)*, pp.23-30 (2002).
2) K. KIM and D. PARK, “Efficient and Scalable Client Clustering for Web Proxy Cache”, *IEICE Transactions on Information and Systems*, Vol.E86-D, No.9 pp.1577-1585 (2003).
3) A. Rowstron and P. Druschel, “Pastry : Scalable,decentralized object location and routing for large-scale peer-to-peer systems”, *Proc. International Conference on Distributed System Platforms(Middleware)*, pp.329-350 (2001).
4) S. Iyer, A. Rowstron and P. Druschel , “Squirrel: a decentralized peer-to-peer web cache”, *Proc. Twenty-first Annual Symposium on Principles of Distributed Computing*, pp. 213-222 (2002).
5) I. Stoica, R. Morris, D. Liben-Nowell, David R. Karger, M. Frans, Kaashoek, F. Dabek, and H. Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for internet applications”, *IEEE/ACM Transactions on Networking*, Vol.11, No.1, pp.17-32 (2003).
6) K. Kim and D. Park , “Efficient and Tailored Resource Management for the P2P Web Caching”, *IEICE Transactions on Information and Systems*, Vol. E90-D, No. 1, pp. 48-57 (2007).
7) M. Jungingerand Y. Lee, “The multi-ring topology-high-performance group communication in peer-to-peer networks”,*Proc. 2nd International Conference on Peer-to-Peer Computing (P2P ’02)*, pp.49-56 (2002).
8) Overlay Weaver: An Overlay Construction Toolkit, http://overlayweaver.sourceforget.net/
9) G. K. Zipf: “Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology”, Addison Wesley (1949)