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

2004－EVA－8　(10)
2004／3／19

Content-Addressable Network における効率的なキャッシング手法の提案

白川 周平　田頭 茂明　藤田 聡

本稿では，P2P ネットワーク上で共有コンテンツの管理／検索機能を実現する Content-Addressable Network において，効率的なインデックスのキャッシング手法を提案し，その有効性を評価する．提案手法では P2P ネットワーク上でのキャッシュの配置方式に着目し，ノード間でキャッシュを効率的に共有することで，インデックスの検索時間の短縮を目指す．具体的には，インデックスをキャッシュする中継ノードをネットワーク内に適切に設定し，検索時に中継ノードを経由するルーティング手法を採用することで，高いキャッシュヒット率を実現している．また提案手法をシミュレーションにより評価し，従来手法と比較して最大で約 30%検索時間を短縮することができた．

Efficient Caching Techniques in Content-Addressable Networks

Syuhei Shirakawa,　Shigeaki Tagashira　and　Satoshi Fujita

In this paper, we propose a caching technique in Content-Addressable Network (CAN) which provides a mechanism for managing and retrieving shared objects distributed over a P2P network by maintaining their indices in a decentralized manner. The proposed technique improves the response time required for retrieving objects by caching indices at the participating nodes. More concretely, the proposed technique locates caches in a deterministic manner to realize an efficient cache location mechanism in the P2P network, and it achieves high hit ratio from the caches, while minimizing the overhead of finding locations. By the result of simulations, we conclude that it can improve the response time by 30% compared with conventional techniques.

1　はじめに

P2P ネットワークにおいて，共有コンテンツのスケーラブルな検索・管理処理を実現する手法として，Content-Addressable Network (CAN)[1]，Chord[2]，Pastry[3] などに代表される分散ハッシュテーブル (DHT) 手法が注目を集めている．CAN では，P2P ネットワーク上に分散している共有コンテンツを，各コンテンツに関連付けられたキーワード毎のインデックスにより管理し，それらすべてのインデックスをネットワークを構成するノード全体で分散して管理している．インデックスの検索時には，検索キーワードのインデックスを持つノードへ，CAN に参加している他のノードを経由して要求を出し，そのインデックスを獲得する．CAN ではこのように P2P ネットワークにおける完全分散型のコンテンツ検索・管理処理を実現するが，検索に時間を要することが問題点として指摘されている [4].

本稿では，情報検索における検索キーワード間の頻度の偏り [9] に着目し，キャッシュを用いて検索頻度の高いキーワードのインデックスを短時間で検索できる手法を提案する．提案手法では，P2P ネットワーク上でのキャッシュの配置に着目し，ノード間での積極的なキャッシュの共有方式を実現する．具体的には，インデックスをキャッシュする中継ノードをネットワーク内に適切に設定し，検索時に中継ノードを経由するルーティング手法を採用することで，高いキャッシュヒット率を実現している．また提

広島大学大学院工学研究科情報工学専攻
〒 739-8527 東広島市鏡山 1-4-1
Graduate School of Engineering, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527 Japan

案手法をシミュレーションにより評価し，従来手法と比較して最大で約 30%検索時間を短縮することができた．

2 CAN

文献 [1] において，P2P ネットワーク上で分散型の共有コンテンツの管理／検索機能を提供する CAN が提案されている．CAN においては，ネットワーク上に分散している共有コンテンツは，インデックスにより管理され，インデックスはシステムを構成するノードに分散して格納される．具体的には，共有コンテンツは設定されたキーワード k から適切なハッシュ関数を用いて，d 次元トーラス状空間（以下ではこの空間をハッシュ空間と呼ぶ）上の座標 p_k にマッピングされ，その座標を管理するノード上に，コンテンツのインデックス（キーワードと，コンテンツを保持するノードのアドレス）が格納される．ここでハッシュ空間の各ノードへの割り当ては，ノードの追加・削除に伴い，動的に行われる．ノードに割り当てられたハッシュ空間の部分空間はゾーンと呼ばれ，隣接ゾーンを管理するノード間では互いにゾーン情報が交換される．隣接していないゾーンを管理するノードとの通信は，隣接ゾーンを経由（ルーティング）することで実現される．具体的には，隣接ノードの中で最も目的座標に近づく隣接ノードと通信を行い，この操作を繰り返すことで，目的座標を管理するノードとの通信を実現する．

コンテンツの検索時には，検索するノードが検索キーワードから座標を算出し，その座標を管理するノードから，対応するインデックスを獲得する．検索ノードはインデックスからキーワードのコンテンツを持つノードのアドレスを取得し，そのノードに対して要求することで，目的のコンテンツを取得することができる．$d = 2$ の場合における検索キーワード”sigeva”の具体的な手順を図 1 に示す．

3 関連研究

本稿では CAN 上にキャッシュを導入することを考え，インデックスをキャッシングすることでインデックスの検索時間の短縮を目指す．特に P2P ネットワーク上でのキャッシュの配置に着目し，ノード間でのキャッシュの積極的な共有方式について考えていく．本章では P2P 上のキャッシング手法について紹介する．

文献 [6] では，P2P ネットワーク上に配置されたコンテンツの複製（キャッシュ）を，P2P ネットワークとは独立した直接通信可能なサーバに管理させ，複製へのアクセスの簡略化／効率化を図っている．この手法は複製へのアクセスを低コストで実現するが，ノード数が増加するとサーバがボトルネックとなりスケーラビリティの点で問題がある．

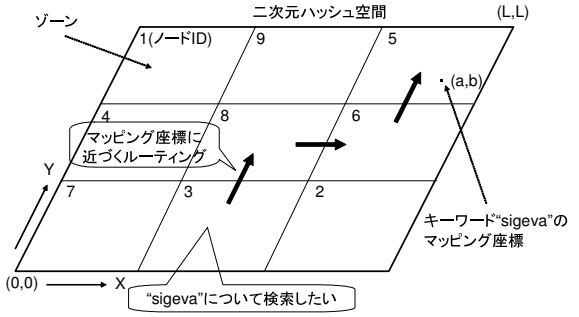

図 1: キーワード”sigeva”のインデックス検索

文献 [1] では，サーバを利用しない CAN 上でのキャッシング手法が提案されている．この手法のことを本稿では CC 手法と呼ぶ．CC 手法では，各ノードが一度検索したインデックスをローカルの記憶領域上にキャッシュし，同一の検索を行う場合にはキャッシュを利用する．またキャッシュを他のノードと共有することで，キャッシュの再利用性を高めている．具体的には，他のノードの検索要求を転送する際，対応するインデックスをキャッシュに保持しているかを確認し，保持しているならば要求を転送せずにキャッシュを利用する．CC 手法はサーバを用いずに実現できるが，ノード間でキャッシュを積極的に共有していないことが問題点としてあげられる．すなわち，従来の CAN における（キャッシュを考慮しない）ルーティングが利用されるために，キャッシュが近隣のノードに存在していても利用されない．

文献 [7] では Probabilistic 手法が提案されている．Probabilistic 手法では “Attenuated Bloom filter” と呼ばれる情報を用いて，キャッシュの所在情報を近隣のノードと共有し，近隣のノード間でキャッシュの効率的な共有を実現している．Attenuated Bloom filter は，Bloom filter を拡張して構成されたキャッシュの所在情報を記憶する．各ノードは，Bloom filter として，l $(l > 0)$ ビットで構成されたビット列を持っており，キャッシュ内のインデックスのキーワードを適切な x 個のハッシュ関数を用いてハッシュ値 m_x $(m_x < l)$ に変換し，Bloom filter 上の m_x ビット目を 1 にセットする．この操作を保持している全てのキャッシュに対して繰り返すことにより，l ビッ

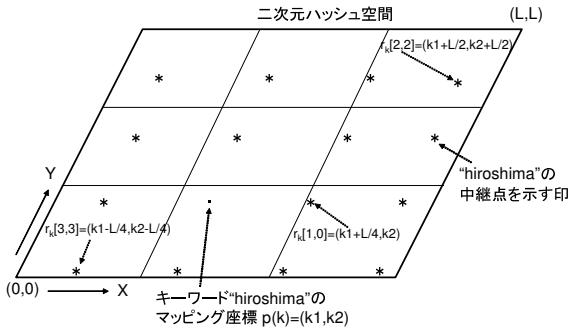

図 2: 提案手法における中継点の設定例（$n = 2$）

トの情報で自身が持つキャッシュの概要を表現することができる．Attenuated Bloom filter は，m 個 (深さ) の Bloom filter を用いて構成される．深さ q $(1 \leq q \leq m)$ の Bloom filter には，q ホップで到達可能な全てのノードが持つ Bloom filter の論理和の結果が格納される．この手法では，ハッシュ関数を用いているために，ハッシュ値の衝突からミスヒットが生じる点と，Attenuated Bloom filter の配布のためのコストが必要な点が問題である．

4 CCPR 手法

4.1 概要

我々は CAN 上のキャッシング手法として，CC 手法を拡張した CCPR 手法を提案する．CCPR 手法では，キャッシュの効果を高めるために，ノード間でキャッシュを効率的に共有する枠組みを提供する．具体的には，各ノードが検索結果のインデックスをキャッシュすることに加えて，キーワード毎に中継ノードと呼ぶノードを設定し，中継ノードにもそのインデックスをキャッシュする．検索時には，中継ノードを経由するルーティングを採用することで，他のノードが検索した同一の要求に対するキャッシュを利用することができる．また，中継ノードを経由するルーティング距離がキャッシングの性能に影響しないように，ハッシュ空間上に中継ノードを適切に設定する．次節からは中継ノードの設定方法とルーティング方法について説明する．

4.2 中継ノードの設定

CCPR 手法では，キーワード毎に中継ノードを適切に設定する．キーワード k の座標 $p_k = (k_1, ..., k_d)$ から算出される座標（中継点）を管理するノードが，そのキーワードについての中継ノードとなる．CCPR 手法では，サイズが等しい部分空間にハッシュ空間を分割し，中継点を各部分空間に一つ配置する．サイズが等しい部分空間に分割することから，CAN の次元数を d とすると，中継点数は 2^{dn} 個（$1 \leq n$）に制限される．これによりハッシュ空間の一辺のサイズを L とすると，ハッシュ空間上の任意の点から最寄りの中継点には $L\sqrt{d}/(2^{n+1})$ 以内で到達できる．

具体的な中継点の設定方法について説明する．キーワード k の中継点の集合を $R_k = \{r_k[i_1, ..., i_d] | (0 \leq i_1, ..., i_d < 2^n)\}$ とすると，$r_k[i_1, ..., i_d]$ は以下のように表される．

$$r_k[i_1, ..., i_d] = (k_1 + L/2^n \times i_1, ..., k_d + L/2^n \times i_d)$$

ただし空間がトーラス状であるため，L を超える座標の場合には L を引くことになる．$i_1 = i_2 .... i_d = 0$ は，p_k になることに注意されたい．2 次元 CAN 上で中継点数を 16 個 $(n = 2)$ とした際の提案手法における中継点の設定例を，図 2 に示す．

4.3 ルーティング

キーワード k の検索ノードは，できるだけルーティング距離を短くして多くの中継点を通過するようにランドマーク点を決定し，ランドマーク点を経由するソースルーティングにより要求を p_k へ送信する．

検索要求には，ランドマーク点としてランドマークテーブル $LM[s](0 \leq s \leq d)$ を保持させ，検索キーワードとともに送信する．ランドマークテーブル作成の具体的な手順は，検索キーワード k をハッシュ関数で変換し，p_k を求める．次に検索ノードから最寄りの中継点を選択し，ランドマークテーブル $LM[0]$ に設定する．$LM[0]$ を決定後，$LM[0]$ から p_k までに通過する中継点を設定するために，$LM[s](1 \leq s \leq d)$ を求める．$LM[s]$ には，$LM[s-1]$ の中継点の i_s を 0 にした中継点が設定される．すなわち，ランドマーク点は次元毎に p_k に近づくように設定されることになる．このとき経由する中継点の総数は，LM[0] における $\Sigma_{e=1}^{d} i_e$ となる．$LM[0] = r_k[2, 4, 2, 4, 2]$ の場合のランドマークテーブルの例を表 1 に示す．

検索要求は検索ノードから送信され，CAN のルーティングで $LM[0]$ に到着する．$LM[0]$ 到着後は $LM[1]$ に送信され，同様に $LM[s]$ 到着後は $LM[s+1]$ に送信され，$LM[d](= p(k))$ に到着するまで繰り返される．CCPR 手法ではこのルーティング上にあるすべてのキャッシュを利用することができる．2 次元 CAN 上でキーワード”sigeva”を検索する場合の例を，図 3 に示す．

表 1: ランドマークテーブル

LM[0]	$r_k[2,4,2,4,2]$
LM[1]	$r_k[0,4,2,4,2]$
LM[2]	$r_k[0,0,2,4,2]$
LM[3]	$r_k[0,0,0,4,2]$
LM[4]	$r_k[0,0,0,0,2]$
LM[5]	$r_k[0,0,0,0,0]$

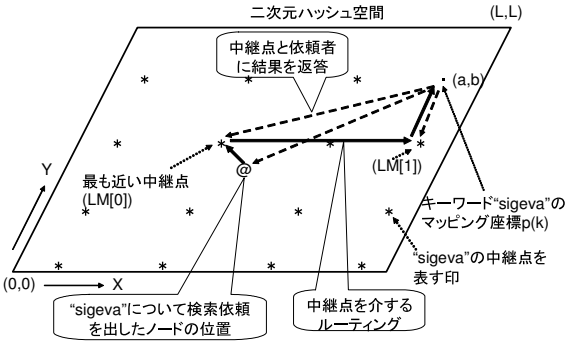

図 3: 提案手法における検索の例

5 評価

本章では CCPR 手法の有用性を示すため，CC 法と Probabilistic 法とを比較対象に用いて，シミュレーションによる評価を行った．なおシミュレーターには NS-2[8] を用いた．

5.1 パラメータ

シミュレーションにおける具体的なパラメータを表 2 に示す．下位層のネットワークトポロジーとしては，全ノードの 1 割がウェブ状のバックボーンネットワークとして構築され，その他のノードはバックボーンネットワークに接続されるものとしている．またシミュレーション中のノードの CAN への動的な参加と離脱については，考慮しないものとする．

事前に全てのノードを，文献 [1] で述べられている手法を用いて CAN に参加させる．参加後ノードは 10 個以内のキーワードをランダムに選択し，各ノードが持つ共有コンテンツとして CAN に登録する．検索の際は検索キーワードをジップの法則（最も人気度の高いキーワードの出現頻度を p とすると，n 番目に人気度の高いキーワードの出現頻度は p/n となる）により選択し，インデックスにアクセスする．文献 [5] で人気度の上位 1%のキーワードについての検索要求が全体の約 20%を占め，上位 10%のキーワードについての検索要求が約 45%を占めることが報告されていることから，より現実に近いシミュレーションをするために，ジップの法則で用いる p を 0.12 に設定した．Probabilistic 手法に用いる Attenuated Bloom filter は，3 個のハッシュ関数，320bit，深さを 2 とした．また，Attenuated Bloom filter を配布する通信量と時間は，本シミュレーションでは考慮しない．

表 2: シミュレーションパラメータ

CAN のハッシュ空間	8192 × 8192
ネットワーク帯域	100Mbps
ノード数	200 〜 4000
全キーワード数	2,336 単語
シミュレーション時間	21,600 秒（6 時間）
各ノードの検索頻度	平均で 360 秒に 1 回
キャッシュ置換法	LFU 方式

5.2 キャッシュサイズの影響

参加ノード数を 2000 に，CCPR 手法の中継ノード数を 16 に固定し，キャッシュサイズを変化させた環境において，インデックスの検索時間を計測した．本シミュレーションでは，キャッシュサイズとしてキーワード数を用いている．

結果を図 4 に示す．CCPR 手法では CC 法や Probabilistic 法と比べ，検索時間が短縮されていることがわかる．キャッシュサイズが 50 の時には Probabilistic 法と比較して，約 20%短縮されている．この理由を示すため，本実験におけるキャッシュヒット率と，ヒット時においてインデックスの獲得までに要したホップ数を計測した．その結果を図 5 と図 6 に示す．CCPR 手法では，他の手法に対し大幅にヒット率が向上している．この結果は CCPR 手法の特徴である中継ノードの効果により，ノード間でのキャッシュの共有が，積極的に行われていることを示している．一方で，CCPR 手法ではヒット時のホップ数が悪化し，どのようなキャッシュサイズでも他の手法に対して 1.5 ホップほど増加している．この理由は，中継ノードを経由するルーティングが原因であると推測される．しかし，検索時間の結果から CCPR 手法のヒット時のホップ数の増加の影響と比べ，他の手法の低ヒット率によるミスヒット時のホップ数の方が大きく影響していることがわかる．

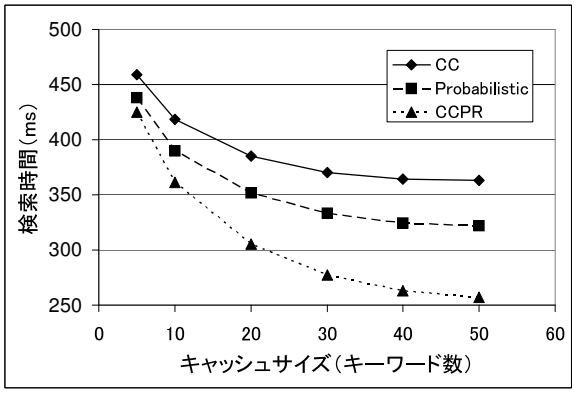

図 4: キャッシュサイズに対する平均検索時間

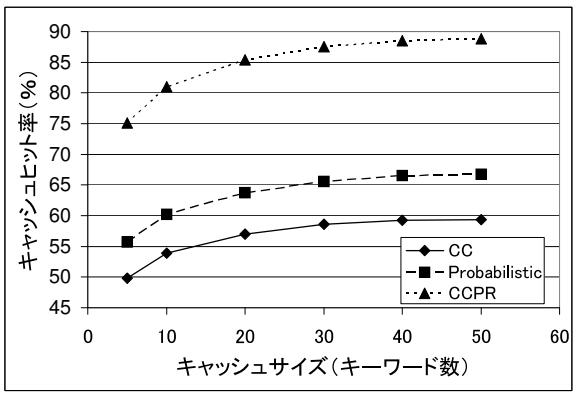

図 5: キャッシュサイズに対するヒット率

5.3 スケーラビリティの評価

参加ノード数の変化に伴う影響を調べるため，キャッシュサイズを 30 に固定し，参加ノード数を 200 台から 4000 台の間で変化させた環境において，各手法の比較を行った．また，CCPR 手法の中継点数が 4，16，64 の場合についても評価した．平均検索時間の結果を図 7 に，キャッシュヒット率を図 8 に，ヒット時の平均ホップ数を図 9 に示す．図中の CCPR 手法の () 内の数字は中継点数を示している．

参加ノード数が増加した場合，CCPR 手法は他の手法より検索時間の短縮率が高くなっている．ノード数が 4000 台の時は Probabilistic 手法に対し，約 30%短縮されている．CCPR 手法ではノード数が 200 から 4000 に増加することで，キャッシュヒット率が 30%近く向上するのに比べて，他の手法ではおよそ 20%しか向上していないことが理由である．ノード数の増加に伴いシステム全体のキャッシュ容量が増加するが，CCPR 手法はキャッシュ容量の増加部分を効果的に利用できているため，このような結果が得

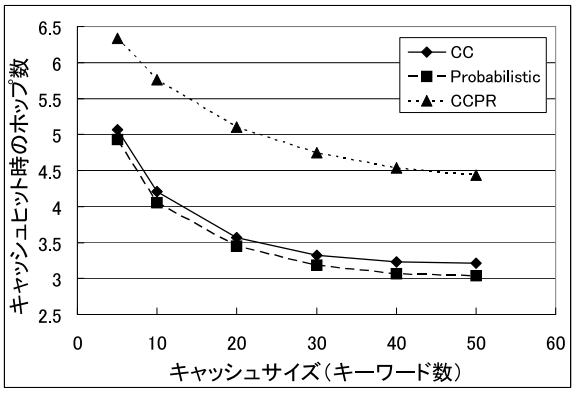

図 6: キャッシュサイズに対するヒット時の平均ホップ数

られたと考えられる．CCPR 手法ではキャッシュヒットまでのホップ数も増加するが，前節で述べたようにミスヒットによる影響の方が大きいため，CCPR 手法の方が良い結果を得られている．

次に，CCPR 手法における中継点数の影響について考察する．中継点数が 16 や 64 の場合では，経由する中継点数が増えるため，キャッシュヒット率が向上している．中継点数が 4 のときと比べて 64 のときでは，約 8%向上している．しかし，中継点数が 64 では，インデックスを管理するノードへ到達するルートが増えるため，近隣の中継点でキャッシュにヒットする可能性が低くなり，キャッシュヒットまでのホップ数は長くなる．すなわち，より遠くの中継点でヒットする可能性が高くなる．一方で，中継点数が 4 の場合では，検索時間が悪化していることがわかる．これは，経由する中継点数が少なくなるためヒット率が低下し，最寄りの中継点までのホップ数が長くなることが原因であると考えられる．以上の理由により，中継点数 16 のときが最も性能が良くなっている．

6 おわりに

本稿では，CAN におけるインデックスのキャッシング手法を提案し，その有効性を評価した．CCPR 手法では，CAN 上でインデックスをキャッシュする中継ノードを設定し，検索時に中継ノードを経由するルーティング手法を採用することで，高いヒット率を実現している．また提案手法をシミュレーションにより評価した．その結果として，中継ノードを経由するホップ数の増加による影響と比べて，キャッシュヒット率の向上による影響が効果的に作用し，検

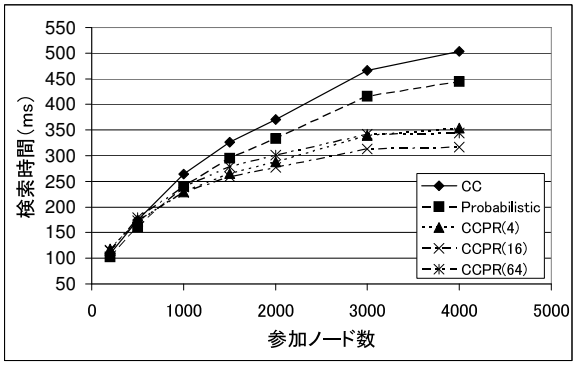

図 7: ノード数を変化させた場合の平均検索時間

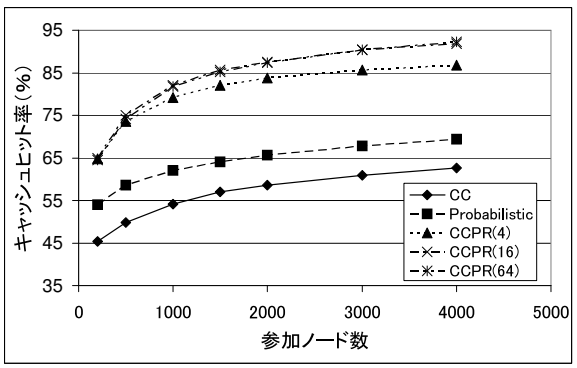

図 8: ノード数を変化させた場合のヒット率

索時間を従来手法と比較して，最大で約 30%短縮することができた．

参考文献

[1] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM '01*, pp.161-172, 2001.

[2] Ion Stoica, Robert Morris, David Karger, M.Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM '01*, pp.149-160, 2001.

[3] Antony Rowstron, Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp.329-350, 2001.

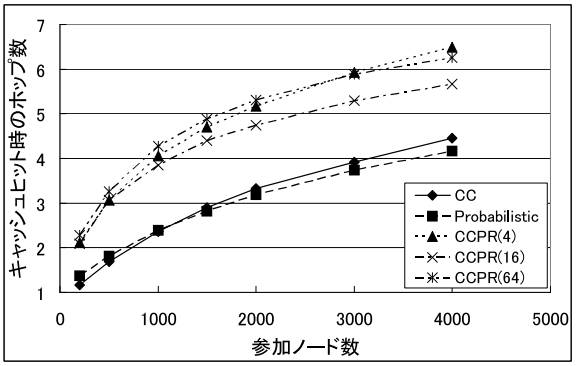

図 9: ノード数を変化させた場合のヒット時の平均ホップ数

[4] Russ Cox, Athicha Muthitacharoen, and Robert T. Morris. Serving DNS using a Peer-to-Peer Lookup Service. In *Electronic Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pp.155-165, 2002.

[5] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of IEEE INFOCOM*, pp.126-134, 1999.

[6] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, C hristopher Wells, and Ben Zhao. OceanStore: An Architecture for Global-scale Persistent Storage. In *Proceedings of ACM ASPLOS*, pp.190-201, 2000.

[7] Sean C. Rhea, and John Kubiatowicz. Probabilistic location and routing. In *Proceedings of INFOCOM*, pp.1248-1257, 2002.

[8] NS-2, http://www.isi.edu/nsnam/.

[9] Matei Ripeanu, and Ian Foster. Mapping the Gnutella Network: Macroscopic Properties of Large-Scale Peer-to-Peer Systems. IEEE Internet Computing, Vol.6 No.1, pp.50-57, 2002.