B 木構造に基づく Bloom フィルタにおけるフィルタの特性に応じた性能評価

佐久間洋[†]　　佐藤文明[††]

P2P における情報検索では、分散ハッシュテーブルや、複数のキーワードでの検索が可能な Bloom フィルタを利用して、情報を検索する方法が従来から研究されている。Bloom フィルタは情報の特徴をビットパターンによって表現するデータ構造であり、OR 演算による情報の結合や、AND 演算よる情報検索に用いられる。先行研究にて構造型 Bloom フィルタにおける類似性に基づく集約コスト削減方式を提案したが、検索要求の伝搬回数が十分に削減できない問題があった。今回、さらに検索効率が向上するように、順序性に着目して B 木構造を構成する方式を提案する。また、Bloom フィルタのサイズ、bit の立て方などの Bloom フィルタの作成方法を考慮しながら、B 木構造に基づく Bloom フィルタにおけるフィルタの特性に応じた性能評価を行う。

Performance Evaluation for Filter Specification of B-Tree Based Bloom Filters

HIROSHI SAKUMA[†]　　FUMIAKI SATO[††]

In information lookup in P2P, a variety of lookup algorithms such as the distributed hash tables (DHT) and Bloom filters have been proposed. The Bloom filter is a data structure where the feature of information contents is expressed by the bit pattern, and it is used to compose information by OR operation and to lookup information by AND operation. Though we proposed have proposed the method to reduce the composition cost of Bloom filters based the similarity of filters, there is a problem the the forwarding cost of the query is not reduced enough. In this paper, therefore, we propose the method of reducing the total forwarding cost of the query based on the order of the Bloom filters. Furthermore, we evaluate the performance of the B-tree based Bloom filters in the different situations such as a size and bit placement of the filters.

1. はじめに

P2P における情報検索では、代表的なものにハイブリッド型とピュア型と呼ばれる方法がある。また、近年では分散ハッシュテーブル[1,2,3,4]を用いた情報検索の方法が研究されている。分散ハッシュテーブルを使った情報検索は、問い合わせをフラッディングするピュア P2P に比べてネットワークの負荷が軽く、ハイブリッド P2P にくらべてサーバの負荷が広く分散できる特徴がある。しかし、分散ハッシュテーブルには、複数のキーワードでの検索がしにくいことや、範囲検索が難しいといった問題点がある。これに対して、複数のキーワードでの検索が可能な Bloom フィルタ[5]を利用した情報検索の方法が従来から研究されている。

フィルタにおける問い合わせの転送方式に分散ハッシュテーブルの一方式である Chord[4]を使った方式[6]や、m分木の B 木に基づく方式[7]が提案されている。

B 木構造に基づく Bloom フィルタの方式では、ノードを配置する方法としてノード ID に基づいて配置する方法をとっている。しかし、この方法では情報検索時に擬陽性発生率が高くなり検索効率が悪くなることや、木構造へのノード参加・脱退時に中間ノードが持つ Bloom フィルタの再構築回数が多くなってしまう。そこで、先行研究[8]では、m分木の B 木に基づく Bloom フィルタにおいて、情報検索の効率化と木構造へのノード参加・脱退時に発生する Bloom フィルタの結合処理を削減する方法として、ノードが持つ Bloom フィルタの類似性に着目して B 木構造を構成する方式を提案した。その結果、従来研究と比べ、情報検索の効率化と木構造へのノード参加・脱退時に発生する Bloom フィルタの結合処理を削減できることが確認できた。しかし、類似性のみだと部分木が不規則に構築されるので、検索要求の伝搬回数が十分に削減できない問題があった。また、従来方式、先行研究ともに Bloom フィルタのサイズ、bit の立て方などの Bloom フィルタの作成方法による B 木構造への影響を評価してこなかった。

本研究では、検索要求の伝搬回数を削減するために Bloom フィルタの順序性に着目して B 木構造を構成する方式を提案する。順序性とは Bloom フィルタのビットの平均的な位置のことであり、この順序性に基づいて木を構成することで、位置のそろった Bloom フィルタを部分木に集める効果がある。また、B 木構造に基づく Bloom フィルタにおけるフィルタの特性に応じた性能をシミュレーションにて評価を行った。

[†] 東邦大学大学院理学研究科
TOHO UNIVERSITY, GRADUATE SCHOOL OF SCIENCE
[††] 東邦大学理学部情報科学科
TOHO UNIVERSITY , FACULTY OF SCIENCE , DEPARTMENT OF INFORMATION SCIENCE.

2. 関連研究

2.1 Bloom フィルタ

Bloomフィルタは、ある要素がサイトに存在することをハッシュ関数と一定のビット列を用いて表現するデータ構造である。

Bloom フィルタを構成するには，まず n ビットのビット列を用意し、全ビットを“0” に初期化する。次に、0 から $n-1$ の値を返す k 個のハッシュ関数により、要素のハッシュ値を求める。そして，各々のハッシュ値に該当する位置のビットを“1” に変える。Bloom フィルタを用いて要素の有無を判断するには、検索したい要素のハッシュ値に対応する位置の Bloom フィルタのビットを調べる。対応する全てのビットが“1” であれば，要素が存在すると判断できる。ただし。ハッシュ値が衝突することにより。要素が存在しないにもかかわらず、要素が存在すると判断する可能性（擬陽性）がある。逆に、要素が存在するにもかかわらず、要素なしと判断することは起こり得ない。Bloom フィルタには、複数の Bloom フィルタの論理和をとることで、それぞれの Bloom フィルタに含まれる全ての要素を表現する Bloom フィルタが得られるという特徴がある。このとき，要素数に関わらず Bloom フィルタのビット長は変化しない。本研究では、Bloom フィルタを用いてコンテンツのキーワードを表現する。Bloom フィルタを使うことで、キーワード数に関わらず同じビット幅でキーワードを表現でき、また複数のコンテンツのキーワードをまとめて表現することが可能となる。図 1 は、コンテンツのキーワードを Bloom フィルタによって表現する例である。

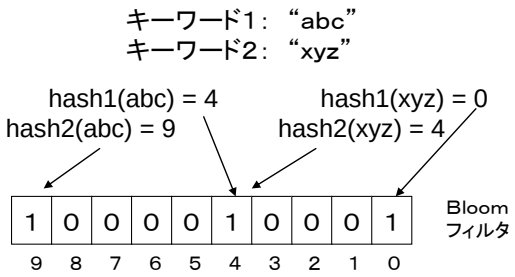

図 1　Bloom フィルタの構成例

2.2 Chord の構造を持つ Bloom フィルタ

通常の分散ハッシュテーブル（DHT）では，複数のキーワードから一つもしくは複数のコンテンツ ID を生成し、それぞれの DHT 型 P2P のプロトコルに従って、コンテンツ ID に対応するノードにコンテンツを追加する。コンテンツを検索するには、検索キーワードからコンテンツ ID を生成し、それぞれの DHT 型 P2P のプロトコルに従って検索を行う。一方、本研究ではコンテンツの追加・削除を行う際に、従来の“コンテンツ ID” を使わない。コンテンツの追加に関して制約はなく、ネットワーク中のどのノードに追加しても良い（ただし負荷分散のため，配置が分散されることが望ましい）。また、コンテンツのキーワードは Bloom フィルタを用いて表現し、DHT 型 P2P のネットワーク構造に従って Bloom フィルタの集約を行うことで、Bloom フィルタの比較による検索を可能にした。

以下、DHT 型 P2P の一つである Chord に基づく従来研究[8]の詳細を説明する。

コンテンツを管理するノードは、管理しているコンテンツの全てのキーワードから、Bloom フィルタを生成する。これにより、ノードが管理しているコンテンツのキーワードを表現する Bloom フィルタができる。これを Node Bloom フィルタ（以下 NBf）と呼ぶことにする．各ノードの NBf を参照することで、キーワード検索を行うことができるようになる。しかし、検索を行う毎に全ノードの NBf を参照するのは効率が悪いため、各ノードの NBf を論理和によって集約することで，DHT 型 P2P の探索経路にどのようなキーワードがあるかを判断できる新たな Bloom フィルタを作成し、検索を効率化する。DHT 型 P2P の構造に従って Bloom フィルタを集約するため、DHT 型 P2P と同様に効率の良い検索が可能となる。Chord では、finger table 内の各ノードについて、それぞれの探索経路の範囲内の全ノードの NBf の論理和をとる。例えば、Node0 は各探索経路の範囲の NBf の論理和から、新たな Bloom フィルタを作成する。こうして得られる Finger Bloom フィルタ（以下 FBf）は、各探索経路の全てのキーワードを含む。FBf を作成するに当たって、各ノードは finger table 内のノードと通信するだけでよく、以下のような処理を行う。ここで、finger[i] は finger table 内の i 番目のノード、FBf[i] は finger[i] についての FBf を表している。

$$\begin{aligned} \mathrm{FBf}[i] &= \mathrm{finger}[i].\mathrm{FBf}[0] + \mathrm{finger}[i].\mathrm{FBf}[1] \\ &+ \cdots \\ &+ \mathrm{finger}[i].\mathrm{FBf}[i-1] \end{aligned}$$

検索では、検索キーワードから検索 Bloom フィルタを生成し、これに従って検索メッセージをルーティングする。各ノードが DHT の構造に従って集約した Bloom フィルタと検索 Bloom フィルタとを比較し、検索 Bloom フィルタに当てはまる Bloom フィルタに対応するノードにのみ検索を転送することで、基礎となる DHT と同様の探索経路を使った検索が可能となる。ただし、探索範囲が重複しないよう、検索メッセージを転送する際に探索範囲を指定する必要がある。

この Chord に基づく Bloom フィルタの方式では、ノード数を N とすると、各ノードが管理する Bloom フィルタ数は $O(\log_2 N)$となる。そのため、各ノードが多くの情報

を保持しなければならないという問題点がある。

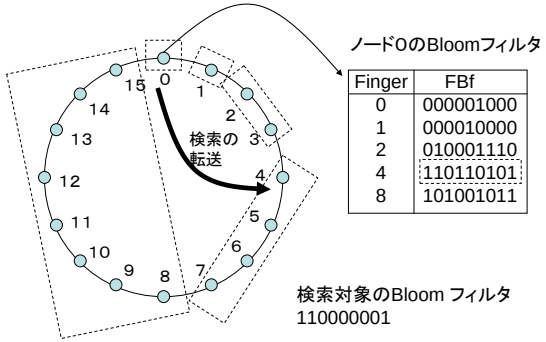

図 2 Chord 構成の Bloom フィルタによる検索例

2.3 B 木構造に基づく Bloom フィルタ

(1) B 木構造

各ノード（P2P インデックス情報を管理している実際のノード）は、分散型のＢ木によって管理されている。Ｂ木におけるノードは、物理的なノードとは異なるため、論理ノードと呼ぶ。Ｂ木における葉ノードは、物理ノードの ID が格納されているものとする。また、木の中間ノードには、木の接続関係（親ノード、子ノードへの分岐の状況を示すノード ID の配列など）や情報のバージョン番号が管理されている。また、根ノードおよび各中間ノードは、下に連結された中間ノードの中の一番小さい ID を分岐の区切りとして、配列に保存して管理することとする。この配列中の一番小さい ID を持つ物理ノードを、代表ノードと呼び実際にその配列を管理する責任を持つものとする。

このＢ木の情報は､参加する代表ノードである物理ノードによって分散管理される。代表ノードである各物理ノードは、B 木の部分情報を持っている。B 木に変更が生じた場合は、他のノードに変更情報を通知することで、全物理ノード間の一貫性を管理している。代表ノードである各物理ノードが持つ部分木の情報は、対応する葉ノードから根ノードまで、木を遡った経路に存在する各論理ノードが持つ情報と、それらに隣接する兄弟ノードが持つ情報である。しかし、他の物理ノードが管理している情報は共有せず、その物理ノードに管理を任せる。

(2) ノード参加・脱退方法

新規の物理ノードの参加方法は、既に参加しているノードのどれかにアクセスして自身の ID を含む参加要求を送付するものとする。アクセスされたノードは、根ノードの代表ノードにアクセスする。要求された根ノードの代表ノードは、要求の ID に基づいて、どの中間ノードに管理されるべきかを判断し、要求を下位の中間ノードに転送する。これを繰り返すことで、管理されるべき最も下位の中間ノードの代表ノードに参加要求が到達する。

もし、空きがない場合、中間ノードはＢ木におけるノードの分割作業を行う。変更情報の通知は、変更が生じた最も上位のノードの代表ノードに依頼して実施してもらう。

既に参加しているノードの脱退方法は、脱退する物理ノードが属する最下位の中間ノードの代表ノードに脱退要求を送付する。変更情報の通知は、ノードの参加処理での通知方法と同様に、変更が生じた最も上位のノードの代表ノードに依頼して実施してもらう。もし要素数が m/2 未満になる場合は、隣接する兄弟ノードから配列の要素を移動する。隣接ノードから配列の要素を移動するとき m/2 未満になる場合は、その隣接ノードと中間ノードを合併する。この合併操作は、必要に応じて根ノードまで遡り、場合によっては最上位の根ノードを削除する操作までを行う。

(3) Bloom フィルタの管理

各物理ノードでは､インデックス情報が管理されおり､各インデックス情報は Bloom フィルタを持っている。各ノードは、ノード内の Bloom フィルタをすべて集約した集約 Bloom フィルタを持っている。また､B 木の構造に基づいて､上位の中間ノードは、下位のノードの Bloom フィルタを集約した集約 Bloom フィルタを持っている｡最上位の根ノードには､すべての Bloom フィルタを集約した全体の集約 Bloom フィルタを持っている。

B 木の情報と同様に、Bloom フィルタの情報も代表ノードである各物理ノードに分散管理されている。管理する Bloom フィルタは対応する葉ノードから根ノードまで、木を遡った経路に存在する各論理ノードが持つ Bloom フィルタと、それらに隣接する兄弟ノードが持つ Bloom フィルタである。しかし、他の物理ノードが管理している Bloom フィルタは共有せず、その物理ノードに管理を任せる。m を分岐数、N をノード数とすると、中間ノードの Bloom フィルタ数は約 O(N/(m-1))となる。代表ノードである各物理ノード数は N/m となるので、代表ノードである各物理ノードが管理する Bloom フィルタ数は約 O(1)となる。

Chord に基づく Bloom フィルタの方式[6]と比べて管理する情報量が少なく、管理コストが抑えられるという利点がある。

(4) **情報の検索方法**

情報の検索では、まず検索用のキーワードから検索用の Bloom フィルタを作成する。検索用 Bloom フィルタを含む検索要求を、B 木のどこかのノードに送付する。検索要求が到達したノードでは、そのノードが保持する B 木の部分情報について、根ノードから比較を開始する。B 木の各ノードに付随する集約 Bloom フィルタと検索用の Bloom フィルタとの AND 演算を実施して、検索用 Bloom フィルタが残る（情報が見つかった）最も下位（つまり、部分木中の葉）のノードを見つける。そのノードが中間ノードであれば、その下の各中間ノードの代表ノードに問い合わせを送る。また、そのノードが葉ノードであれば、対応する物理ノードに直接検索要求を送付する。
転送された検索要求を受け取った中間ノードの代表ノードは、その中間ノード以下のノードに対して検索要求に含まれる Bloom フィルタを使って照合を実施する。そして、最も下位のノードに到達するまで、繰り返し検索要求を行う。

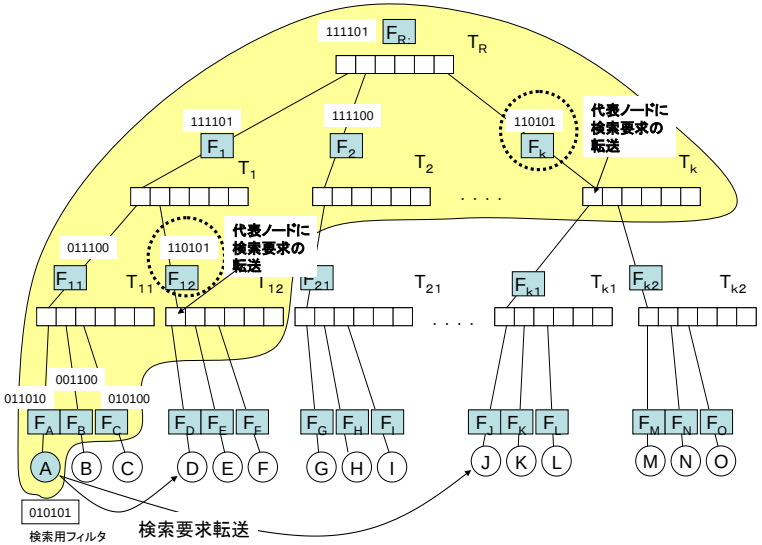

図 2 ノード A への検索要求の転送例

3. 提案方式

2.3 節で述べた B 木構造に基づく Bloom フィルタの構成は、情報検索時に擬陽性発生率が高くなり検索効率が悪くなることや、木構造へのノード参加・脱退時に中間ノードが持つ Bloom フィルタの再構築回数が多くなってしまう。

先行研究[8]では、m 分木の B 木に基づく Bloom フィルタにおいて、情報検索の効率化と木構造へのノード参加・脱退時に発生する Bloom フィルタの結合処理を削減する方法として、ノードが持つ Bloom フィルタの類似性に着目して B 木構造を構成する方式を提案した。Bloom フィルタの類似性を導入することで、似ている Bloom フィルタを持つノードを近くに配置できるようになる。そうすることにより、情報検索時には擬陽性発生率が低下し、検索効率が上がり、また、ノード参加・脱退時には中間ノードが持つ Bloom フィルタの再構築回数を少なくすることができる。また、今回はさらに情報検索の効率が向上するよう Bloom フィルタの順序性に着目して B 木構造を構成する方式を提案する。

3.1 類似性の判定方法

類似性とは、Bloom フィルタ同士がどのくらい似ているかを示すものである。類似性は、整合性と順序性に基づいている。

整合性は、Bloom フィルタ同士で同じ位置にどのくらいビットが立っているかを表す数値である。ある Bloom フィルタと比較対象のいくつかの Bloom フィルタを比べて同じ位置のビットが立っている数が多い方が Bloom フィルタ同士が類似していると判断する。整合性の判定方法は、ある Bloom フィルタと、比較対象のいくつかの Bloom フィルタとの AND 演算をとることにより、立ったビット数を整合性とし、数値が大きい方が Bloom フィルタ同士の類似性が高いと判断する。

整合性が同じだった場合には、Bloom フィルタを結合する際になるべく立たせるビットを少なくさせる方が Bloom フィルタの構造的に良いため、Bloom フィルタのビットが立っていない位置の整合性を判定する方法をとる。それぞれの Bloom フィルタのビットをすべて反転させ AND 演算をとることにより、立ったビット数を整合性とし、数値が大きい方が Bloom フィルタ同士の類似性が高いと判断する。

ビットを反転させて出した整合性も同じだった場合には、Bloom フィルタの順序性を算出し、順序性が近い Bloom フィルタを類似性が高いと判断する。順序性とは Bloom フィルタの立っているビットの平均的な位置を表す数値のことである。左よりのビット列であれば順序性の数値は大きく、右よりのビット列であれ順序性の数値は小さくなる。算出方法は Bloom フィルタのビットが立っている位置の数値を全て足し、立っているビット数で割ることにより算出する。

図 3 は整合性の判定方法の例、図 4 は順序性の判定方法の例である。

図 3 では、追加するノードが持つ Bloom フィルタをフィルタ A、比較対象の Bloom フィルタをそれぞれフィルタ X，フィルタ Y、フィルタ Z としている。フィルタ A とフィルタ X，フィルタ Y、フィルタ Z との整合性はそれぞれ、2、3、1 である。この場合フィルタ A との類似性が一番高いフィルタはフィルタ Y ということになる。

図 4 では、追加するノードが持つ Bloom フィルタと比較対象の Bloom フィルタ X、Y の順序性の比較判定の例である。追加するノードが持つ Bloom フィルタの順序性と比較対象の Bloom フィルタの順序性を比較し、順序性の近い Bloom フィルタ Y を類似性が高いと判断する。

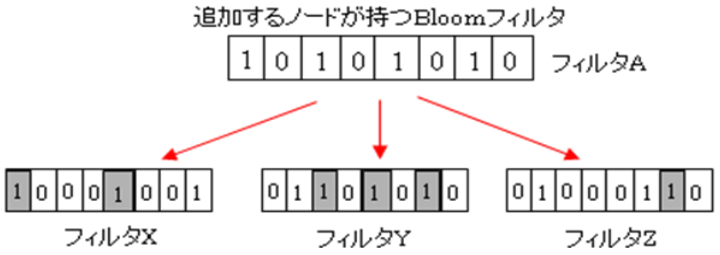

図 3　整合性の比較判定の例

図 4　順序性の比較判定の例

3.2 新規ノード参加方法

新規ノードの参加方法は、参加するノードが持っている Bloom フィルタを根が持つ Bloom フィルタから木のレベルごとに順に類似性を比べていき、類似性の高い方へと下っていく。最下位の中間ノードまでこれを繰り返していき、最終的に参加する位置を決める。

また、B 木構造のノード配置方法において順序性を使う。B 木構造の部分木のノードの並び方に導入することで、順序性が大きい Bloom フィルタを持つノードは B 木構造の左側に配置されやすくなり、順序性が小さい Bloom フィルタを持つノードは B 木構造の右側に配置されやすくなる。この方法により情報検索の効率化を図る。

4. シミュレーション

4.1 シミュレーション方法

先行研究[8]で提案した類似性に基づく構造型 Bloom フィルタの構成と今回提案した Bloom フィルタの順序性に着目して B 木構造の構成において、検索要求の平均伝搬回数の比較を行う。

Bloom フィルタのサイズ、bit の立て方などの Bloom フィルタの作成方法を考慮して Bloom フィルタの特性に応じた性能評価を行う。このシミュレーションでは、従来研究[7] であるノード ID によってノードが配置された B 木構造に基づく Bloom フィルタの構成と今回の提案方式を用い、ノード参加・脱退時に発生する中間ノードが持つ Bloom フィルタの再構築回数と検索要求の平均伝搬回数の評価を行う。Bloom フィルタの再構築回数とは、ノードが参加または脱退した際に発生する中間ノードが持つ Bloom フィルタの結合処理の回数のことである。検索要求の伝搬回数とは、検索要求が根ノードから目的の情報を持つノードまでに通る中間ノードの総数であり、検索要求の散らばりも計測している。平均伝搬回数とは何回か計測した伝搬回数の平均値である。

シミュレーションは C 言語で書いた独自のプログラムで評価を行う。また、シミュレーションは以下の表 1 のシミュレーション条件の下で行う。

表 1 シミュレーション条件

木の分岐数	5
ノード数	10,000、20,000、50,000、100,000
検索回数	1000 回
参加・脱退回数	2000 回、2000 回
Bloom フィルタのサイズ	64bit、128bit、256bit、512bit

4.2 シミュレーション結果

図 5 は先行研究[8]で提案した類似性に基づく構造型 Bloom フィルタの構成と今回提案した Bloom フィルタの順序性に着目して B 木構造の構成において、検索要求の平均伝搬回数の比較である。Bloom フィルタのサイズは 128bit、ハッシュ関数の数は 4 で評価した。

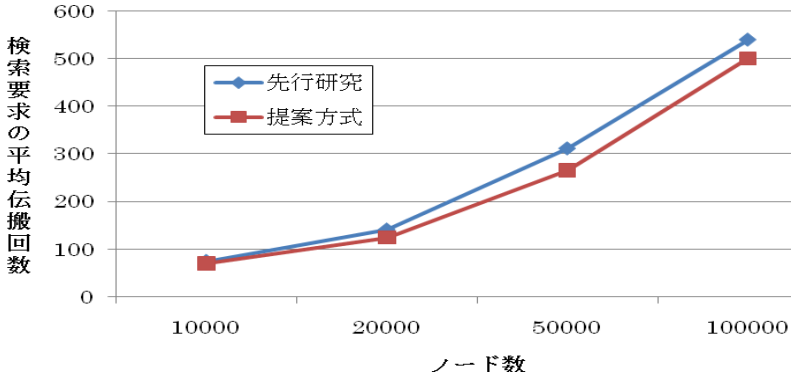

図 5 検索要求の平均伝搬回数

図 5 より、検索要求の平均伝搬回数は先行研究よりも今回提案した順序性に着目した B 木構造の構成の方がいずれのノード数の場合でも 7%から 14%ほど少なくなることが確認できた。

図 6 は従来方式での Bloom フィルタ作成時のハッシュ関数の数に応じた中間ノードが持つ Bloom フィルタの再構築回数と検索要求の平均伝搬回数を示し、図 7 は提案方式の Bloom フィルタ作成時のハッシュ関数の数に応じた中間ノードが持つ Bloom フィルタの再構築回数と検索要求の平均伝搬回数を示す。いずれもノード数 10,000 として評価した。その結果、検索要求の平均伝搬回数については、どちらの方式でもハッシュ関数の数が 3 の場合、良好となった。一方、Bloom フィルタの再構築回については、従来方式ではハッシュ関数の数が多くなればなるほど再構築回数は減少したが、提案方式では Bloom フィルタのサイズによって最適なハッシュ関数の数は変わることが分かった。

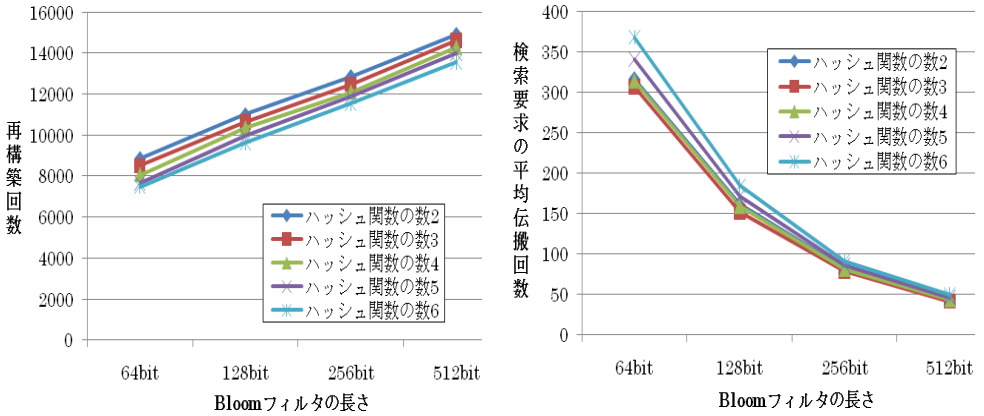

図 6 ハッシュ関数の数に応じた再構築回数と検索要求の平均伝搬回数(従来方式)

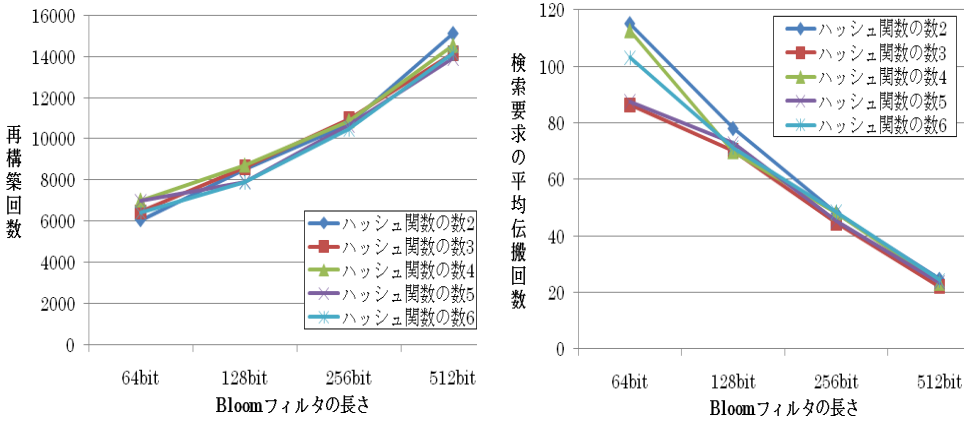

図 7 ハッシュ関数の数に応じた再構築回数と検索要求の平均伝搬回数(提案方式)

図 8 は提案方式においてのノード数、Bloom フィルタのサイズ、検索要求の伝搬回数をそれぞれ比較したものと、ノード数、ハッシュ関数の数、検索要求の伝搬回数をそれぞれ比較したものである。その結果、ノード数に関わらず Bloom フィルタのサイ

ズを大きくすると基本的に検索要求の平均伝搬回数は少なくなることが分かった。また、ノード数 10,000 の場合はハッシュ関数の数 3、ノード数 100,000 の場合はハッシュ関数 6 のときに検索要求の平均伝搬回数は最小となり、ノード数が増えるごとに検索要求の平均伝搬回数が最小となるハッシュ関数の数は増えることが分かった。しかし、Bloom フィルタのサイズを大きくするとノード参加・脱退時に Bloom フィルタの再構築回数は増加することが分かっているので、環境に応じて Bloom フィルタのサイズを決める必要があると思われる。

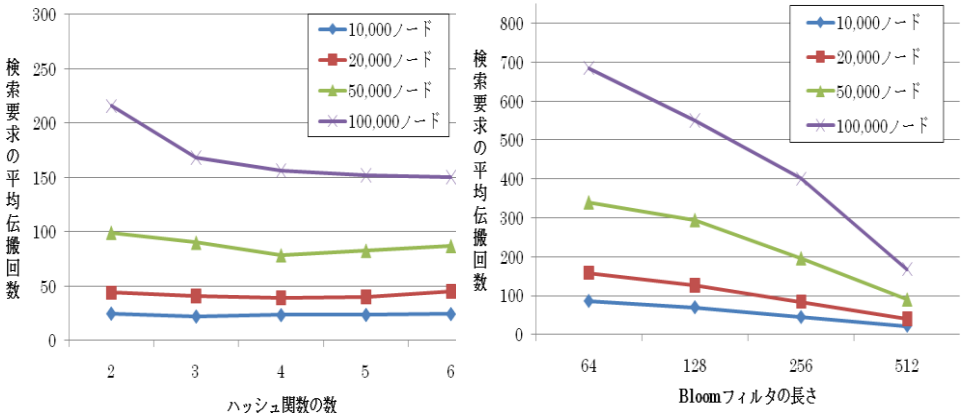

図 8 ノード数ごとの検索要求の平均伝搬回数

図 9 は B 木構造に基づく Bloom フィルタにおいて、ノード参加・脱退時に発生する中間ノードが持つ Bloom フィルタの再構築の回数を従来方式・提案方式それぞれにおいて Bloom フィルタのサイズごとに計測し、比較した。ノード数は 10,000、ハッシュ関数の数は 3 とした。図 10 は B 木構造に基づく Bloom フィルタにおいて、検索要求を出した時の検索要求の平均伝搬回数を従来方式・提案方式それぞれにおいて Bloom フィルタのサイズごとに計測し、比較した。ノード数は 10,000、ハッシュ関数の数は 3 とした。

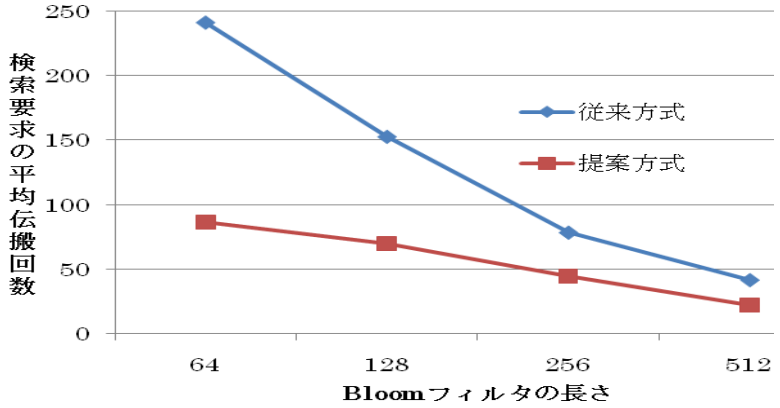

図 9 Bloom フィルタの再構築回数

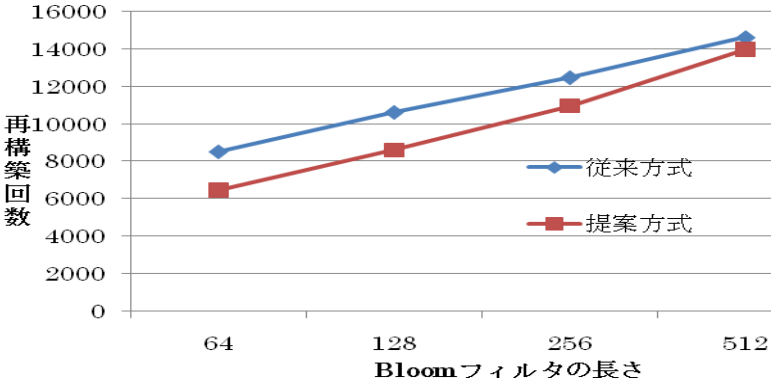

図 10 検索要求の平均伝搬回数

図 9 より、従来研究と提案方式において、いずれの Bloom フィルタのサイズでもノード参加・脱退時に発生する中間ノードが持つ Bloom フィルタの再構築回数が 8%から 25%ほど少なくなることが確認できた。また、Bloom フィルタのサイズを大きくするとノード参加・脱退時に Bloom フィルタの再構築回数は多くなってしまうことが分

かった。

図 10 より、従来研究と提案方式において、いずれの Bloom フィルタのサイズでも情報を検索する際に中間ノードへの検索要求の伝搬回数が 47%から 64%ほど少なくなることが確認できた。これは、上位レベルのノードは多くの Bloom フィルタを集約しており、情報検索時に擬陽性発生率が高くなり、検索要求の伝搬回数が多くなってしまうので、ノードを Bloom フィルタの類似性を考慮して配置することにより、その擬陽性発生率を抑えられるということである。また、Bloom フィルタのサイズを大きくすると検索要求の伝搬回数が少なくなることが分かった。

5. まとめ

本研究では、先行研究[8]で提案した類似性に基づく構造型 Bloom フィルタの構成と今回提案した Bloom フィルタの順序性に着目して B 木構造の構成において、検索要求の平均伝搬回数の比較を行った。その結果、検索要求の平均伝搬回数は先行研究よりも今回提案した順序性に着目した B 木構造の構成の方がいずれのノード数の場合でも 7%から 14%ほど少なくなることが確認できた。

また、Bloom フィルタのサイズ、bit の立て方などの Bloom フィルタの作成方法を考慮して Bloom フィルタの特性に応じた性能評価を行った。その結果、B 木構造基づく Bloom フィルタにおいて Bloom フィルタを作成する際にハッシュ関数の数に応じてノード参加・脱退時に発生する中間ノードが持つ Bloom フィルタの再構築回数と検索要求の伝搬回数が変わってくることが分かった。再構築回数は、従来方式[7]ではハッシュ関数の数が多くなればなるほど減少したが、提案方式では Bloom フィルタのサイズによって最適なハッシュ関数の数は変わった。一方、検索要求の伝搬回数は、ノード数 10,000 の従来方式、提案方式ともに Bloom フィルタのサイズに関わらず基本的にハッシュ関数の数が 3 の場合、良好となることが分かった。また、基本的にどちらの方式でも Bloom フィルタのサイズを大きくすると検索要求の伝搬回数は削減できるが、逆に、ノード参加・脱退時に発生する中間ノードが持つ Bloom フィルタの再構築回数は多くなってしまうことが分かった。ノード数ごとの検索要求の平均伝搬回数は、ノード数 10,000 の場合はハッシュ関数の数 3、ノード数 100,000 の場合はハッシュ関数 6 のときに検索要求の平均伝搬回数は最小となり、ノード数が増えるごとに検索要求の平均伝搬回数が最小となるハッシュ関数の数は増えることが分かった。

シミュレーションの結果より、B 木構造に基づく Bloom フィルタにおいては、ノードが頻繁に参加脱退をするような環境であれば Bloom フィルタの再構築回数が少なくなるような Bloom フィルタ構成方法、ノード参加脱退が少なく検索効率が重要な環境では検索要求の伝搬回数が少なくなる Bloom フィルタの構成方法にするといったことが望ましいと思われる。

今後の課題としては、ノード参加・脱退時に発生する Bloom フィルタの再構築回数と検索要求の伝搬回数が共に少なくなるような Bloom フィルタの作成方法、整合性・順序性の利用方法、ノード参加・脱退方法を考えることである。

参考文献

1) Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In Proceedings of the ACM SIGCOMM 2001 Technical Conference, San Diego, CA, USA, August 2001.

2) Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pages 329-350, November 2001.

3) Petar Maymounkov and David Mazieres. Kademlia: A Peer-to-peer Information Systems Based on the XOR Metric. In Proceedings of the IPTPS 2002, pages: 53-65, Boston, March 2002.

4) Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In Proceedings of the ACM SIGCOMM 2001, San Diego, CA, USA, August 2001.

5) B. Bloom. “Space/Time Tradeoffs in Hash Coding with Allowable Errors.” Communications of the ACM 13:7 (1970), pp.422-426,July1970

6) 佐藤一帆,松本倫子,吉田紀彦,”複数キーワード検索に対応した分散ハッシュ型 P2P ネットワーク”,FIT2007, pp 437-440, November2007.

7) 若林繁寿, 佐藤文明, “Bloom Filters Based on the B-Tree” 社団法人 情報処理学会 研究報告 2008-DPS-137 (9) pp43-48, November 2008

8) 佐久間洋, 佐藤文明, “構造型 Bloom フィルタにおける類似性に基づく集約コスト削減方式の提案” マルチメディア，分散，協調とモバイル(DICOMO2010)シンポジウム, pp. 2023-2031, July2010.