類似性に基づく構造型Bloomフィルタの構成方式

佐久間洋†　佐藤文明†
東邦大学理学部情報科学科

1 はじめに

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[1]。

Bloomフィルタ[2]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演算による情報検索に用いられる。

我々はB木構造を持つBloomフィルタによる情報検索を提案してきた[3]が、Bloomフィルタの類似性については考慮してこなかった。そこで、今回Bloomフィルタの類似性に着目して、木構造へのノードの追加削除をする際の結合処理を削減するB木構造の構成方法について提案する。

2 関連研究

2.1 Bloomフィルタとは

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。

2.2 B木構造を持つBloomフィルタ

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。

このB木の情報は、参加する物理ノードによって分散管理される。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、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。

なお、根ノード、及び各中間ノードは、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。

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要求を転送する。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。

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。

3 提案方法

3.1 概要

このB木構造を持つBloomフィルタの構成において、ノードの参加・脱退時に中間ノードが持つBloomフィルタが変更された場合にフィルタの変更を伝搬するコストを削減するノードの参加・脱退方法を提案する。

3.1 フィルタ変更の伝搬コスト削減方法

フィルタの変更の伝搬を減らす方法として、類似性の高いフィルタを持つノードをグループ化する方法を提案する。

Composition Method of Structured Bloom Filter Based on Similarity.
†Hiroshi Sakuma, †Fumiaki Sato ・Toho University , Faculty of Science , Department of Information Science.

新規ノードの追加方法は、根が持つ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、最終的に追加する場所を決める。

類似性の判定方法は、追加するノードが持つBloom フィルタと、比較対象のフィルタとの AND 演算をとることにより、立ったビット数を類似性とし、比較していく。類似性が同じだった場合には、Bloom フィルタのビットをすべて反転させ AND 演算をして立ったビット数が多い方を類似性が高いと判断する。

新規ノード追加時に中間ノードが分岐の上限を越える場合は、中間ノードを分割し、ノードを 2 つのグループに分ける必要がある。分割方法は、m分木の場合、追加するノードが持っている Bloom フィルタと類似性の高いフィルタを持っているノードを(m+1)/2-1 個のノードを選び、選ばれたノードとそれ以外のノードとでグループに分ける。

4 シミュレーション

4.1 シミュレーション方法

Chord[4]型の Bloom フィルタと B 木構造の検索要求の平均ホップ数を比較した。

また、既存のB木構造を持つ Bloom フィルタの構成と今回提案した類似性に基づく構造型 Bloom フィルタの構成において、ノード追加・脱退時に発生するフィルタの再構築の回数を比較した。

- Chord 型と B 木構造においての検索要求がホップされる回数を計測した。
- ノード数を 1,2,5,10 万ノードで固定し、1000 回のノード追加・脱退を繰り返し、再構築回数を計測した。

4.2 シミュレーション結果

図 1 は Chord と B 木構造の検索要求にかかる平均ホップ数を比較した結果である。これは、B木構造の方が Chord よりも平均ホップ数が少ないことが分かる。

図２は既存のB木構造を持つ Bloom フィルタの構成と今回提案した類似性に基づく構造型 Bloom フィルタの構成において、ノード追加・脱退時に発生する Bloom フィルタの再構築の回数を比較した結果である。ノード追加・脱退時ともに Bloom フィルタの再構築回数を削減できたことが分かる。

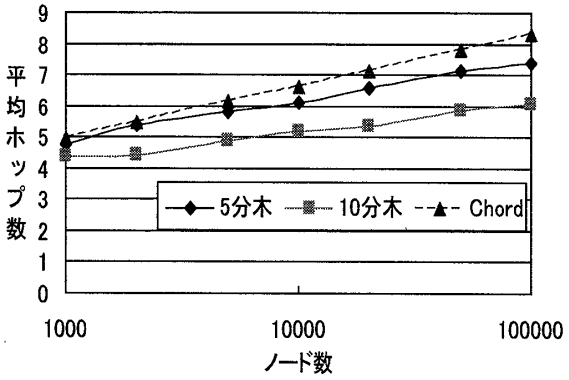

図１　検索要求にかかる平均ホップ数

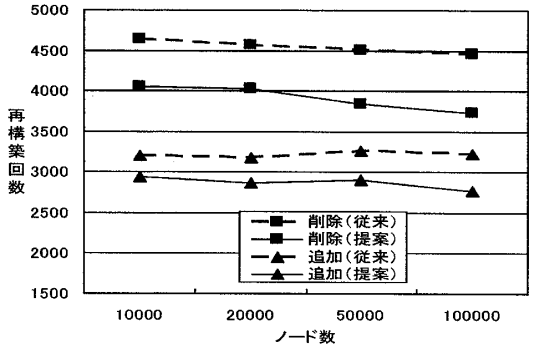

図２　フィルタの再構築回数

5 おわりに

本稿では Bloom フィルタ間に類似性を定義し、新規ノードを追加する際、そのノードが持つフィルタと類似性の高いフィルタを持つノード付近へ挿入する方法を提案した。その結果、中間ノードがその下に持つそれぞれのノードは類似性が高いフィルタを持っている。これは、ノードをノード ID に基づいて追加する場合よりも検索要求を減らすことができ、また、再構築のコストも減らすことができた。

今後の課題は、類似性の大小に加えて、フィルタに順序性を導入することである。順序性を導入することで検索要求が送られる領域を限定することが可能になると思われる。

参考文献

[1] Andrei Broder, Michael Mitzenmacher, Network Applications of Bloom Filters: A Survey, Internet Mathematics, pp.636-646,(2002).

[2] B. Bloom. “Space/Time Tradeoffs in Hash Coding with Allowable Errors.” Communications of the ACM 13:7 pp422-426,(1970).

[3] 若林繁寿, 佐藤文明, “Bllom 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, (2008)

[4] 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