2003－DBS－131　(56)
2003／7／17

P2P 環境におけるシグネチャを用いたオブジェクト検索方式の検討

松下　亮[†]，　北川　博之[††]，　石川　佳治[††]

概要

近年，計算機の高性能・低価格化とネットワークインフラの発達により P2P 技術が注目されている．グローバルな索引等を持たない P2P 環境では，オブジェクトの効率的検索をどのように実現するかが問題となる．我々の研究グループでは，オブジェクトの特徴をシグネチャとして表現し，分散型シグネチャを用いることで，効率的かつ柔軟なオブジェクト検索を実現する方式を提案してきた．本研究ではより詳細な検討を行い，メッセージ数と応答時間を考慮したオブジェクト検索および追加処理方式を提案する．さらに人工的データと実データを用いて，各処理方式による違いをシミュレーション実験により評価検討する．

On Signature-based Object Retrieval in Peer-to-Peer Environments

Ryo MATSUSHITA[†], Hiroyuki KITAGAWA[††], Yoshiharu ISHIKAWA[††]

abstract

Peer-to-peer (P2P) technology has attracted a lot of attention in recent years. Efficient object retrieval is an important research issue in P2P environments, especially in those without centralized global indices. We proposed a novel object retrieval method using distributed frame sliced signatures. In this paper, we propose object retrieval and registration strategies that take the number of messages and response times into consideration, and evaluate their effectiveness with simulation experiments using synthetic and real data sets.

1　はじめに

近年，計算機の高性能化とネットワークインフラの発達により，Peer-to-Peer (P2P) 技術が注目されている．P2P では各計算機が peer node(ノード) となり，大規模な分散ネットワークを構築する．P2P ネットワークの形態は，クライアント・サーバシステムを融合させたハイブリッド P2P 型と，完全な分散環境であるピュア P2P 型に分類される．ハイブリッド P2P 型では，ある種のサービスを提供するために特定のサーバが存在する．しかし，多数のノードがそのサーバのサービスを要求した場合には，サーバがボトルネックとなる．一方，ピュア P2P 型では各ノードが自律的に動作する．このため拡張性に富み，ボトルネックのない処理を実現することができる．しかし，ピュア P2P 型ではグローバルなインデックス等をサーバに持つことができないため，一般に情報の共有は容易ではない．代表的なピュア P2P 型のシステムとして，ファイル共有システム Gnutella[4] が挙げられる．Gnutella ではブロードキャストを用いて，周辺のノードを巡回する方法で検索を行うため，オブジェクト検索時における通信コストが大きな問題となる．この問題を解決するため，Chord[12], P-Grid[1]，CAN[11], Tapestry[5] 等の手法が提案されている．しかし，これらの手法では，オブジェクト ID というキーを用いたオブジェクト検索のみが考慮されており，オブジェクトの持つ種々の特徴量による検索を直接行うことはで

[†]筑波大学理工学研究科
Master's Program in Science and Engineering, University of Tsukuba
[††]筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

きない．

我々はこれまでの研究 [9] で，オブジェクトの特徴をシグネチャ[2] [6] として表現し，基本アーキテクチャとなる Chord の枠組みの上にシグネチャを用いた検索機構を構築した．これにより，ピュア P2P 型環境における効率的なオブジェクト検索手法を提案することができた．これまでは単一のオブジェクト検索および追加処理手順のみを検討対象としてきた．しかし，実際には他の処理手順も考え得る．そこで本研究では，本手法のより詳細な検討を行うため，メッセージ数と応答時間を考慮した複数の処理手順を検討対象とする．また，評価実験では，人工的に生成したランダムなデータに加えて，ODP[10] が提供する実データを用いた場合のシミュレーション実験を行う．

以下では，まず 2 章で関連研究について述べる．次に 3 章で本手法で用いている Chord アーキテクチャについて説明する．さらに，4 章でシグネチャの説明をし，5 章で本手法の基本方式について述べる．6 章で本研究が提案する処理手順について説明し，7 章で各処理手順に対する評価実験について述べる．最後にまとめと今後の課題を示す．

2 関連研究

P2P 環境における検索処理を効率化する関連研究として，研究 [13] がある．この研究では，Gnutella の幅優先探索と，Freenet[3] の深さ優先探索を組合せ，メッセージ数と検索の応答時間のトレードオフを実現する．この研究では，幅優先探索と深さ優先探索との組み合わせの方式をいくつか挙げている．これらの方式は，ブロードキャストのポリシーの違いに基づくものであり，各方式に対して実験，評価を行っている．しかし，いずれの方式でもノードを順次巡回することで，対象オブジェクトを探索することに変わりはない．

本手法では P2P 環境におけるオブジェクト検索を実現するためにシグネチャを用いるが，並列処理を用いてシグネチャの照合処理を効率化する研究 [7] もある．この研究では，シグネチャを分割しすべての計算機に均等に割り当て，照合処理を並列化することで効率化を図るような処理方式を提案している．しかし，シグネチャの照合処理が全ノードで発生することや，シグネチャの分割や割当てが静的であることにより，この方式を P2P 環境に適応させることは非常に困難である．

3 Chord アーキテクチャ

Chord では，ネットワーク空間全体が **ID サークル**という円状の仮想空間として定義され，すべてのノードとすべてのデータオブジェクトはこの ID サークル上に配置される (図 1)．Chord のネットワーク空間の大きさは $scale$ で表現され，最大 2^{scale} 個のノードから構成される．Chord ではノードおよびデータオブジェクトを ID サークル上に均等に配置するため，ハッシュ関数を用いる．これによりノードに対して $scale$ ビットの**ノード ID**(nid)，データオブジェクトに対して $scale$ ビットの**オブジェクト ID**(oid) をそれぞれ与える．各ノードはノード ID を基に ID サークル上に配置される．また各データオブジェクトは，オブジェクト ID から時計回りに ID サークルを辿り，適当なノードへ割り当てられる．

各ノードは実際に割り当てられたオブジェクト ID のリスト，ルーティング情報，前後に位置するノード情報 ($successor$，$predecessor$) を保持している．ルーティング情報は，検索対象のオブジェクト ID のオブジェクトが当該ノードに存在しない場合に，次にその問合せをフォワードすべきノードの情報を含む．この情報は $scale$ 個存在し，各 $interval$ に含まれるオブジェクト ID のオブジェクトを検索する際に次にどのノードに検索要求をフォワードすべきかが示されている．ただし，$interval$ とは次式で定義される区間のことである．

$$interval = [(nid + 2^{k-1}) mod\ 2^{scale}, (nid + 2^k) mod\ 2^{scale}) \quad (1 \leq k \leq scale)$$

このように，各 $interval$ の大きさは時計回りに順に 2^{k-1} となっている．これらのルーティング情報を持つことで，各ノードは ID サークル全体

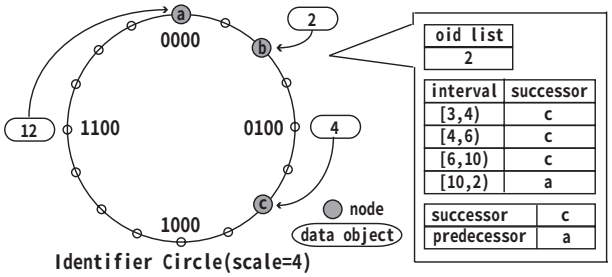

図 1: Chord アーキテクチャ

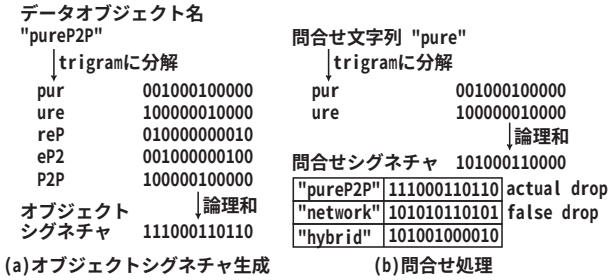

図 2: シグネチャによる部分一致検索

をカバーすることになり，任意の問合せに対して適切なルーティング処理を行うことができる．

4 シグネチャ

シグネチャは，個々のデータオブジェクトから生成される固定長のビット列であり，オブジェクトの特徴量を表現するものである．特徴量をビット列という単純な表現方法に変換することで，特定の特徴量の存在の有無を容易に判定でき，多様な特徴量によるオブジェクト検索が可能である．

まず，各データオブジェクトから生成される**オブジェクトシグネチャ**について述べる．図 2(a) は，データオブジェクト名から生成した *trigram* を特徴量とした場合の例を示している．オブジェクトシグネチャの生成にはスーパーインポーズドコーディングを用いている．これは各特徴量をハッシングしてシグネチャ長の**要素シグネチャ**を生成し，さらに各要素シグネチャの論理和をオブジェクトシグネチャとするものである．なお，シグネチャに含まれる'1' の立っているビットの数のことをシグネチャのウェイトと呼ぶ．

問合せに関しては，オブジェクトシグネチャと同様にして**問合せシグネチャ**を作成する (図 2(b))．オブジェクトシグネチャと問合せシグネチャの各ビットの論理積をとったものが，問合せシグネチャと一致する場合に，問合せ条件を満たす解の候補となる．この解の候補のことをドロップといい，この中で実際に正解となるものを**アクチュアルドロップ**，そうでないものを**フォールスドロップ**という．この判定処理のことをフォールスドロップレゾリューションという．

5 基本方式

本手法では，P2P ネットワークのノード上に，シグネチャ情報を分散配置することで，多様な特徴量に基づくオブジェクト検索の実現を計る．本手法は，P2P 環境上でデータの効率的検索を実現する枠組みの上に構築するものであり，1 章で述べた Chord，P-Grid 等のいずれの枠組みを用いても実現することが可能である．本章は，シグネチャ情報を含むインデックスエントリの配置と，インデックスエントリとの照合による検索について説明する．

検索対象のデータオブジェクトはユーザが任意のノードに配置し，シグネチャ情報を含むインデックスエントリを分散配置する．インデックスエントリの配置処理，および検索処理は Chord の枠組みを利用する．各データオブジェクトはノード単位で管理されるため，データオブジェクトのノード内での ID とそれを格納するノード ID のペアが，データオブジェクトを一意に決定するキーとなる．Chord のオブジェクト ID(*oid*) と本手法のデータオブジェクトの ID を明確に区別するため，本手法におけるノード内でのデータオブジェクトの ID のことを**ローカル ID**(*lid*) と呼ぶ．

5.1 インデックスエントリ

シグネチャ情報の分散配置を行う上での前処理について説明する．まず，データオブジェクトから生成したオブジェクトシグネチャを図 3 に示すように分割フレーム数 *slice* 個の**フレームシグネチャ**に分割する [8]．特に，*slice* がシグネチャ長と一致する場合を**ビットスライス構成**と呼ぶ．次に，フレーム番号をバイナリ表現に変

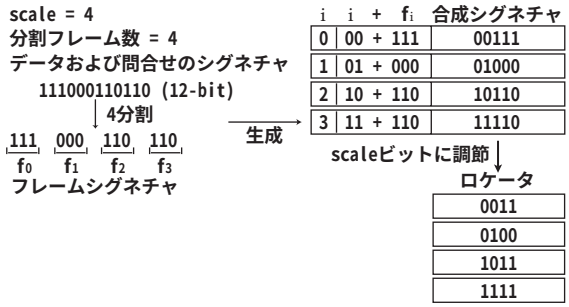

図 3: ロケータ生成

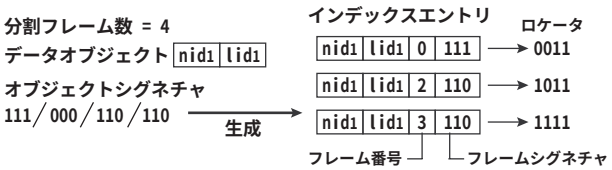

図 4: インデックスエントリ生成

換したビット列と当該フレームシグネチャを結合し，合成シグネチャを得る．

次に**インデックスエントリ**を生成する (図 4)．インデックスエントリは，各フレームシグネチャに対して生成する．ただし，すべてのビットが'0' であるフレームシグネチャに対しては，インデックスエントリは生成しない．インデックスエントリは，当該データオブジェクトのローカル ID，それを格納したノード ID，フレーム番号，およびそのフレームシグネチャから構成される．

5.2 ロケータとインデックスエントリの配置

インデックスエントリの ID サークル上への配置のため，合成シグネチャから長さが $scale$ ビットの**ロケータ**を生成する．ロケータは検索処理の際も利用する．ロケータの生成方法は次の通りである．

[**ケース 1**] 合成シグネチャ長が $scale$ の場合，合成シグネチャ自身をロケータとする．

[**ケース 2**] 合成シグネチャ長が $scale$ より大きい場合，先頭から $scale$ 番目までのプレフィックスをロケータとする．

[**ケース 3**] 合成シグネチャ長が $scale$ より小さい場合，'0' を合成シグネチャに追加することで長さを $scale$ とし，ロケータとする．

あるデータオブジェクトが追加された場合に

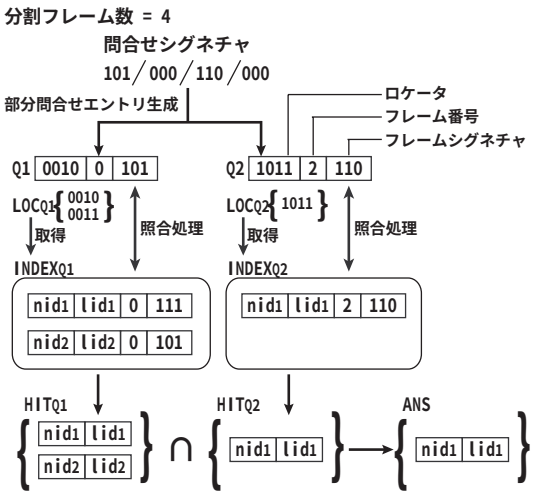

図 5: オブジェクト検索

は，対応するインデックスエントリを生成し，さらにロケータを生成する．各インデックスエントリは 3 章で述べた Chord のアルゴリズムに従い，ロケータをオブジェクト ID とみなして，適当なノードへ配置する．以下では，あるロケータ Loc により配置されているインデックスエントリ集合のことを Loc の**配置インデックスエントリ集合**と呼ぶ．

5.3 オブジェクト検索

オブジェクト検索時は，問合せ条件として与えられた特徴量から，問合せシグネチャ，フレームシグネチャ，合成シグネチャ，ロケータを順次生成する．ただし，フレームシグネチャがすべて'0' で構成されるものに対してはロケータを生成しない．さらに，これらより**部分問合せエントリ**を生成する．部分問合せエントリはロケータ，フレーム番号，フレームシグネチャから構成される．

部分問合せエントリを用いた検索処理の基本方式については既に [9] で述べたので，ここでは例を用いて説明する (図 5)．まず，問合せから部分問合せエントリ集合 $\{Q_1, Q_2\}$ が生成される．次に，各部分問合せエントリのロケータから**検索対象ロケータ集合** LOC_{Q_1}，LOC_{Q_2} を生成する．検索対象ロケータ集合は，ロケータの中にあるフレームシグネチャ中の'0' を'0' または'1' としたすべてのビット列の集合である (したがって次式を満たす．検索対象ロケータ集合の各要素

$\wedge^*$ ロケータ = ロケータ). 検索対象ロケータ集合に含まれるロケータをもつインデックスエントリに対して照合処理を行う必要がある．このようにして，照合対象となるインデックスエントリ集合 $INDEX_{Q_1}$，$INDEX_{Q_2}$ を取得する．さらに部分問合せエントリと照合処理を行い，以下の条件を満たすインデックスエントリを選択し，解候補集合に加える．ただし解候補集合とは，ノード ID とローカル ID のペアを要素とする集合である．

[**条件 1**] 部分問合せエントリ中のフレーム番号 = インデックスエントリ中のフレーム番号．

[**条件 2**] 部分問合せエントリ中のフレームシグネチャ $\wedge$ インデックスエントリ中のフレームシグネチャ = 部分問合せエントリ中のフレームシグネチャ．

すべての検索対象ロケータ集合について解候補集合 HIT_{Q_1}，HIT_{Q_2} の取得処理が終了した時点で，当該部分問合せエントリに関する処理が終了する．最後に，各解候補集合の積集合をとり，最終的な解集合 ANS を得る．

6 処理手順

既に述べたように，インデックスエントリの配置や検索には Chord のアルゴリズムをベースとして用いるが，具体的な処理手順にはいくつかのものが考えられる．一般にオブジェクト検索や追加を行う場合の要求として，メッセージ数を小さくし，通信コストを抑えたいといった要求や，ある程度のメッセージ数を許容し，応答時間を小さくしたいといった要求が考えられる．そこで本研究では，メッセージ数と応答時間に着目した処理手順を検討する．特にここでは，逐次的に処理を行う順次処理と，同時に複数の処理を行う並列処理を取り入れた処理手順について考察を行う．

5 章で述べたように，本手法における検索および追加処理では，シグネチャをフレーム分割し，フレームシグネチャ単位で対象となるロケータを生成する．各処理は生成されたロケータ単位で行われるため，ロケータの処理順序がメッセージ数と応答時間に深く関係してくる．本研究では，(a) メッセージ数を重視した処理手順，(b) 応答時間を重視した処理手順，(c) 中間的処理手順，以上 3 つの処理手順について検討する．これらの処理手順は，検索と追加処理の両方に適用することが可能である．

(a) メッセージ数を重視した処理手順

メッセージ数を重視した処理手順は順次処理のみを行う処理手順のことであり，ノード間でのメッセージのやり取りを効率的に行うことができる．このため，他の処理手順と比較した場合，メッセージ数を小さく抑えることが可能である．しかし応答時間を考えた場合では，順次処理を行うことにより応答時間が大きくなる．

(b) 応答時間を重視した処理手順

応答時間を重視した処理手順は並列処理を導入した処理手順のことである．並列化が可能である処理では常に並列処理を行うため，処理が終了するまでの応答時間を小さくすることが可能である．しかし，並列処理を実行させるために，各ノード間でメッセージをやり取りする必要があり，処理にかかるメッセージ数は大きくなると考えられる．

(c) 中間的処理手順

中間的処理手順は順次処理と並列処理を混合させた処理手順である．本手法では，フレームシグネチャ単位で検索時の解候補集合の取得やインデックスエントリの配置処理を行っている．このため，各フレームシグネチャ単位で並列処理を実行し，当該フレームシグネチャのロケータ単位で順次処理を行う．この処理手順は (a) よりも，応答時間に関して効率的であり，(b) よりもメッセージ数に関して効率的であると考えられる．したがって，(a) と (b) の中間的な処理手順である．

6.1 オブジェクト追加

図 6 がオブジェクト追加時の各処理手順を表している．追加時の場合，フレームシグネチャ単位で生成されるロケータは多くとも 1 つであるため，(b) 応答時間を重視した処理手順と (c) 中間的処理手順は等価である．

(a) メッセージ数を重視した処理手順

図 6(a) のように，配置するインデックスエント

*'$\wedge$' はビット論理積を表す

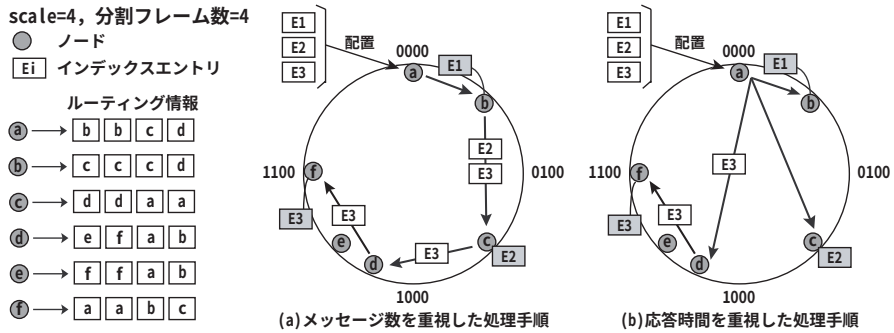

図 6: 処理手順 (オブジェクト追加)

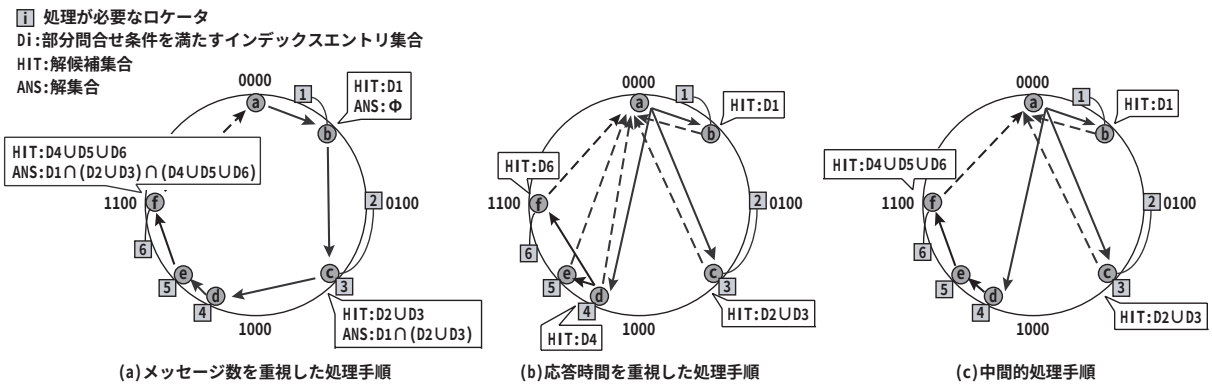

図 7: 処理手順 (オブジェクト検索)

リ集合を順次巡回することで，適当なノードへ各インデックスエントリが配置される．

(b) 応答時間を重視した処理手順

各インデックスエントリごとに並列処理が行われるため，図 6(b) のように配置処理が実行される．

6.2 オブジェクト検索

図 7 が検索時の各処理手順を表している．この図では，すべての問合せはノード a から開始されるものと仮定している．また，実線矢印は処理順序を表しており，破線矢印は解集合および解候補集合を開始ノードへ返すことを表している．

(a) メッセージ数を重視した処理手順

図 7(a) のように順次巡回することで，配置インデックスエントリ集合の取得，照合処理による解候補集合の取得，解集合の絞り込み処理が各ノードで行われる．最終的に開始ノードに解集合 ANS が返され，検索は終了する．

(b) 応答時間を重視した処理手順

検索対象ロケータ集合中の各ロケータの配置インデックスエントリ集合との照合処理を並列処理を用いて行う (図 7(b))．この処理手順では，各ノードから得られた解候補集合を直接開始ノードへ返す．開始ノードは部分問合せエントリ単位で解候補集合を計算し，最終的な解集合の絞り込み処理を行う．

(c) 中間的処理手順

フレームシグネチャ単位で並列処理を行い，順次処理により解候補集合を獲得する (図 7(c))．したがって，開始ノードでは受け取った各解候補集合の積集合をとり，解集合を得る．

7 評価実験

シミュレーションに基づく各処理手順の評価実験を行った．実験は各データオブジェクトに対し，人工的にランダムなシグネチャを生成する方法と，ODP が提供する実データを用いてシグネチャを生成する方法で行っている．ODP データは ASCII 文字で構成されるテキストデータであり，各データの平均文字列長はおよそ 126 文字である．本実験では，特徴量として $bigram$ を

表 1: 主な実験パラメタ

項目	値
$scale$	10
ノード数	128
ノード当たりのデータオブジェクト数	500
データオブジェクトの特徴量の数	164
問合せの特徴量の数	2
シグネチャ長	2^{10}(=1024)
シグネチャのウェイト	512

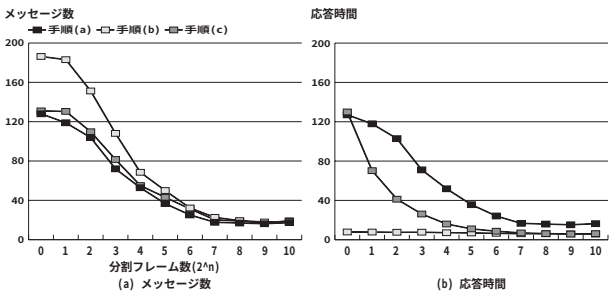

図 8: オブジェクト検索 (人工的データ)

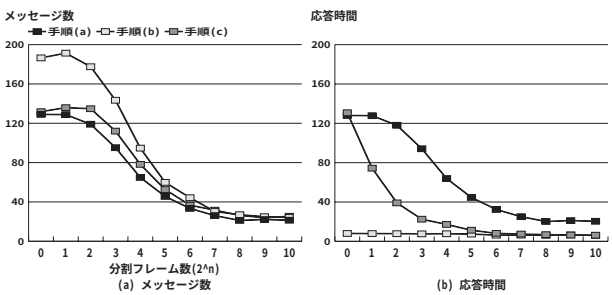

図 9: オブジェクト検索 (ODP データ)

用い，オブジェクトシグネチャを生成する．

実験時の主なパラメタを表 1 に示す．データオブジェクトの特徴量の数に関しては，シグネチャ長が 2^{10} の場合で，特徴量を 1 個与えた問合せに対するフォールスドロップ確率が約 5%となるように定めた．応答時間の測定では，ノード間で，あるメッセージを転送するために必要な転送時間を単位時間とする．その他の照合処理などにかかる処理時間は，メッセージの転送時間と比較した場合に非常に小さくなると考えられるため，考慮しない．

7.1 オブジェクト検索

オブジェクト検索時について，各処理手順の違いを検証するため，評価実験を行った．図 8 が人工的データを用いた場合，図 9 が ODP データを用いた場合の実験結果である．測定値は問合せを 20 回行った平均値である．まず，人工的データを用いた場合と ODP データを用いた場合に関しては，いずれの方法についても同様な曲線を描いており，各処理手順がもつ基本性質はデータの違いには依らないことを表している．

次に，各処理手順の違いについて説明する．全体的な比較では，手順 (a) が最もメッセージ数が小さくなっており，手順 (b) が最も応答時間が小さくなっていることが分かる．特に手順 (b) は，どの分割フレーム数の場合でも応答時間は非常に小さい．しかし，分割フレーム数が 2^{10} 付近では，いずれの処理手順についてもほとんど差はないことを確認できる．

7.2 オブジェクト追加

オブジェクト追加時についても同様の実験を行った．図 10 が人工的データを用いた場合，図 11 が ODP データを用いた場合の実験結果である．測定値はオブジェクト追加を 20 回行った平均値である．この実験についても，人工的データと ODP データを用いた場合では同様な曲線を描いている．分割フレーム数が多い場合に，ODP データを用いた方がメッセージ数が小さくなっているのは，シグネチャのウェイトが関係していると考えられる．ODP データの平均文字列長は 126 文字であり，$bigram$ による特徴量の数としては 125 となる．このため，シグネチャのウェ

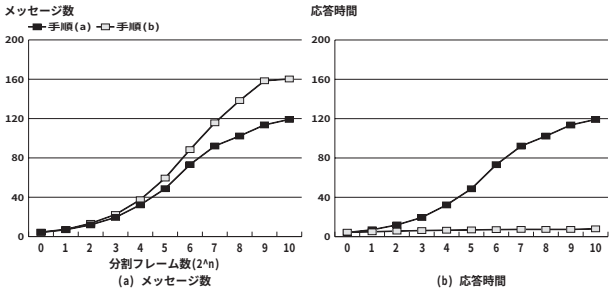

図 10: オブジェクト追加 (人工的データ)

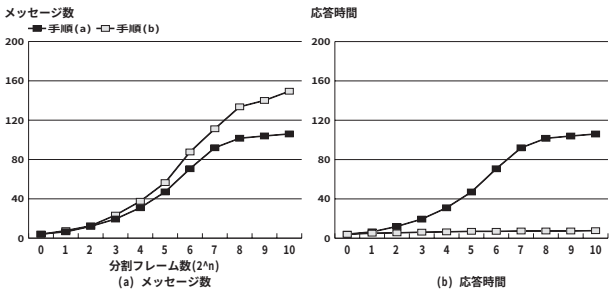

図 11: オブジェクト追加 (ODP データ)

イトが 512 よりも小さくなり，配置するインデックスエントリ数も小さくなっていると考えられるからである．各処理手順の違いについて，手順 (a) では手順 (b) と比べメッセージ数は小さい．一方，手順 (b) は，メッセージ数は大きくなるものの，応答時間は非常に小さくなっていることがわかった．

8 おわりに

本研究では，我々の研究グループが提案してきた P2P 環境におけるシグネチャを用いたオブジェクト検索方式に関して，メッセージ数と応答時間を考慮した処理手順という側面から検討を行った．また，人工的データと ODP が提供する実データを用いて実験を行い，各処理手順の基本的性質を明らかにした．

今後の課題として，複数の問合せを同時に処理する場合のシステム動作の検証，より大規模なネットワーク環境での評価実験，実際の計算機環境における検索時間，更新時間等の計測がある．

謝辞

本研究の一部は，日本学術振興会科学研究費萌芽研究 (15650011)，基盤研究 (B)(15300027)，若手研究 (B)(14780316) による．

参考文献

[1] Karl Aberer, P-Grid: A Self-Organizing Access Structure for P2P Information Systems, CoopIS 2001, LNCS 2172, pp. 179-194, 2001.

[2] Christos Faloutsos, Signature files: Design and Performance Comparison of Some Signature Extraction Methods, Proc. ACM SIGMOD 1985, pp. 63-82, 1985.

[3] Freenet website. http://freenet.sourceforge.net/.

[4] Gnutella website. http://www.gnutella.com/.

[5] Kirsten Hildrum, John D. Kubiatowicz, Satish Rao, and Ben Y.Zhao, Distributed Object Location in a Dynamic Network, SPAA'02, pp. 41-52, 2002.

[6] 石川 佳治, 北川 博之, 大保 信夫, シグネチャファイルによる集合値検索のコスト評価, 情報処理学会論文誌, Vol.36, No.2, pp. 383-395, 1995.

[7] Zheng Lin, Concurrent Frame Signature Files, Kluwer Academic Publishers, Distribued and Parallel Databases Vol.1, pp. 231-249, 1993.

[8] Zheng Lin, and Christos Faloutsos, Frame-Sliced Signature Files, IEEE TKDE Vol.4, No.3, pp. 281-289, 1992.

[9] 松下 亮, 北川 博之, 石川 佳治, P2P 環境におけるシグネチャを用いたオブジェクト検索, 電子情報通信学会第 14 回データ工学ワークショップ (DEWS2003), 2003.

[10] Open Directory Project website. http://dmoz.org/.

[11] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, A Scalable Content-Addressable Network, SIGCOMM'01, pp. 161-172, 2001.

[12] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, SIGCOMM'01, pp. 149-160, 2001.

[13] Beverly Yang and Hector Garcia-Molina, Improving Search in Peer-to-Peer Networks, ICDCS'02, 2002.