P2P 環境におけるシグネチャを用いたオブジェクト検索機構の設計と実装

松下　亮† 須藤　雅則†† 北川　博之††† 石川　佳治†††

† 筑波大学理工学研究科　〒 305–8573 茨城県つくば市天王台 1–1–1
†† 筑波大学第三学群情報学類　〒 305–8573 茨城県つくば市天王台 1–1–1
††† 筑波大学電子・情報工学系　〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †ryo@kde.is.tsukuba.ac.jp, ††stu@kde.is.tsukuba.ac.jp , †††{kitagawa,ishikawa}@is.tsukuba.ac.jp

あらまし　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は，オブジェクトの特徴をシグネチャとして表現することで，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．しかし，これまではシミュレーションに基づく評価実験のみしか行っておらず，実計算機環境を対象とした場合の検討が不十分であった．そこで本研究では，提案方式を採り入れた P2P フレームワークの設計と実装を行う．本フレームワークは，P2P アプリケーションを作成するための基本フレームワークである JXTA を用いる．また，実データを用いて評価実験を行い，実計算機環境における提案方式の有効性を示す．
キーワード　P2P ネットワーク，情報検索，シグネチャファイル

Design and Implementation of Signature-based Object Retrieval in Peer-to-Peer Environments

Ryo MATSUSHITA†, Masanori SUTO††, Hiroyuki KITAGAWA†††, and Yoshiharu ISHIKAWA†††

† Master's Program in Science and Engineering, University of Tsukuba
†† College of Information Sciences, University of Tsukuba
††† Institute of Information Sciences and Electronics, University of Tsukuba
E-mail: †ryo@kde.is.tsukuba.ac.jp, ††stu@kde.is.tsukuba.ac.jp , †††{kitagawa,ishikawa}@is.tsukuba.ac.jp

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 indeces. We proposed a novel object retrieval method using distributed frame sliced signatures. Although, we have evaluated the method with simulation experiments, evaluation in real computer environments was not yet done. In this paper, we design and implement a P2P framework adopting our proposed method. Our framework uses JXTA, which is a fundamental framework for developing P2P applications. We evaluate effectiveness for our method using real data sets in real compouter environments.
Key words　P2P Network, Information Retrieval, Signature Files

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 では各端末がノードとなり，自律的に動作することで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，ハイブリッド P2P 型とピュア P2P 型に分類される．本研究は，拡張性に富みボトルネックのない処理を実現できるピュア P2P 型を対象とする．ピュア P2P 型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 [2] 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，データ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，DHT(Distributed Hash Table) を用いた Chord [10]，CAN [9]，Tapestry [3]，P-Grid [1] といった

手法が提案されている．しかし，これらではキー値による完全一致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きない．

我々はこれまでの研究 [6] [7] で，オブジェクトの特徴量をシグネチャとして表現し，DHT を用いた手法の枠組みの上に，シグネチャを利用した検索機構を構築した．これにより，ピュア P2P 環境における効率的なオブジェクト検索方式を提案することができた．またシミュレーションによる評価実験を行うことで，提案方式の有効性を示した．しかし，これまでは実計算機環境を対象とした評価実験を行っていなかった．そこで本研究では，提案方式を採り入れた P2P フレームワークの設計と実装を行う．本フレームワークは，P2P アプリケーションを作成するための基本フレームワークである JXTA [8] を用いる．さらに，実データとして CNN のニュース記事を用いた評価実験を行い，実計算機環境における本提案方式の有効性を示す．

以下では，まず 2. で関連研究について述べる．次に 3. でシグネチャの説明をし，4. で本手法の基本方式について述べる．5. で本フレームワークの設計と実装について述べ，6. で本フレームワークの評価実験を行う．最後にまとめと今後の課題を述べる．

2. 関 連 研 究

既に述べたように，キー値 (key:長さ k のビット列) に基づく効率的な検索を実現するための方法として，DHT を用いた手法が提案されている．代表的な手法には，Chord [10]，CAN [9]，Tapestry [3]，P-Grid [1] 等がある．これらはネットワーク上に効率的な探索のための構造を導入し，適切なルーティング処理によりオブジェクト検索処理の効率化を図る．ここで，DHT を用いた手法の登録処理と検索処理を以下のように定義し，これらを DHT の基本関数と呼ぶことにする．

- 登録 (key, オブジェクト);
- 検索 (key);

オブジェクト登録時は，key に基づき対象オブジェクトの登録ノードを自動的に決定し登録する．オブジェクト検索時は，登録時と同様に key を用いることで適当なノードに問合せ，適合するオブジェクトを取得する．また，このような key による問合せに対して必ず結果を返すことを保証している．DHT を用いた手法は，これらの処理を $O(log(N))$ に準じたメッセージコストで処理できる．

また，DHT の実装として GISP [4] がある．これは JXTA を用いて実装を行っている．この実装は DHT の基本関数にそれぞれ対応する API を提供している．GISP については 5.1.2 で詳しく説明する．

3. 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．図 1(a) は，データオブジェクトから生成した語 ($word$) 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 F の**要素シグネチャ**を生成し，さらにそれらの論理和をオブジェクトシグネチャとするものである．したがって，生成されたオブジェクトシグネチャは，当該データオブジェクトの持つ特徴量を全て表すことになる．シグネチャでは，1 の立っているビット位置によってこれを表現する．

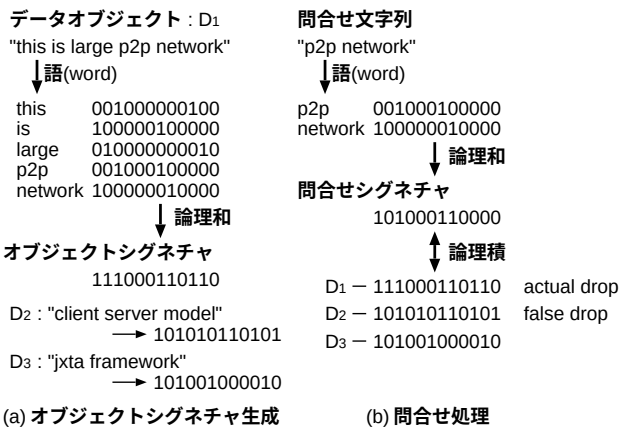


図 1　シグネチャによる部分一致検索

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．オブジェクト検索では，問合せシグネチャの 1 の立っているビット位置がオブジェクトシグネチャでも 1 であるかを照合することで，当該オブジェクトが問合せの持つ全ての特徴量を含む可能性があるかを判定する．したがって，オブジェクトシグネチャと問合せシグネチャ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そのオブジェクトは問合せ条件を満たす解の候補となる．この解の候補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**といい，この判定処理のことをフォールスドロップリゾリューションという (図 1(b))．また，解の候補がフォールスドロップとなる確率をフォールスドロップ確率という．

4. 基 本 方 式

提案方式では，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．本方式は，DHT を用いた効率的処理を実現する枠組みの上に構築するものであり，Chord 等のいずれの枠組みを用いても実現できる．本章はシグネチャ情報を含むインデックスエントリの登録処理と，インデックスエントリとの照合による検索処理について説明する．

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登録し，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登録および検索は，DHT の基本関数を利用する．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 である**ローカル ID**(lid) 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．

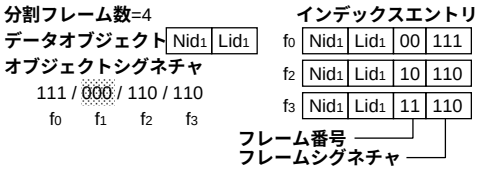


図 2　インデックスエントリ生成

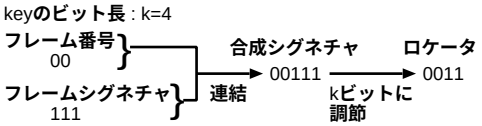


図 3　ロケータ生成

4.1　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 2 に示すように分割フレーム数 m 個の**フレームシグネチャ**に分割する [5]．特に，m 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**インデックスエントリ**を生成する．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．

4.2　ロケータとインデックスエントリの登録

インデックスエントリを適当なノードへ登録するため，フレーム番号をバイナリ表現に変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 (図 3)．合成シグネチャから長さが k 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以下である．

- **ケース 1**

合成シグネチャ長が k 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 k より大きい場合，先頭から k 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 k 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 k 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 2. で述べた DHT の基本関数に従い，ロケータを key とみなして，適当なノードへ登録する．

このロケータ生成方法を用いることにより，任意の分割フレーム数，およびシグネチャ長に対してロケータを生成することができる．また，このようなロケータを用いてインデックスエントリを配置することで，各ノードに対して均等にインデックスエントリを配置することができる．

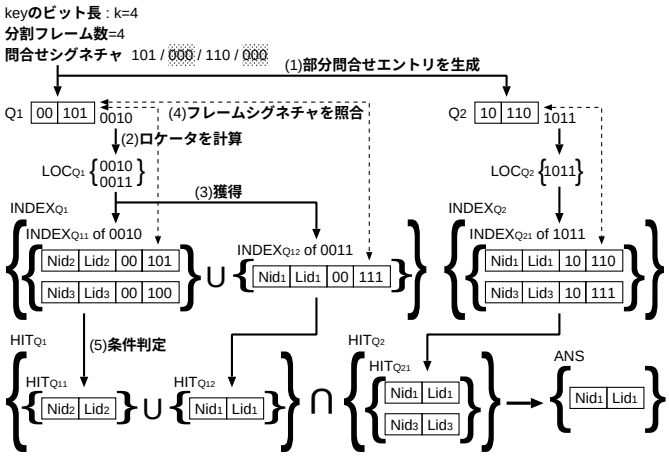


図 4　オブジェクト検索

4.3　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．問合せはこれらの部分問合せエントリと分散配置されたインデックスエントリ集合を照合することで行われる．

検索処理について例を用いて説明する (図 4)．まず，(1) 問合せから部分問合せエントリ集合 $\{Q_1, Q_2\}$ を生成する．次に，(2) 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 $\wedge^{(注1)}$ 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，(3) 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 (4) 部分問合せエントリと照合処理を行い，(5) 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．

(注1)：'$\wedge$' はビット論理積を表す

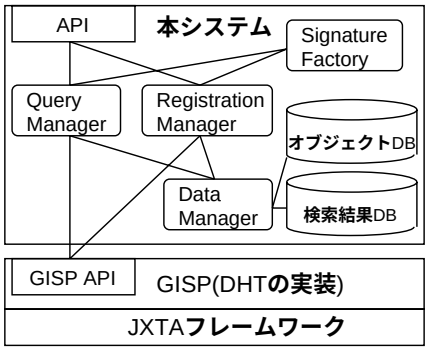


図 5　アーキテクチャ

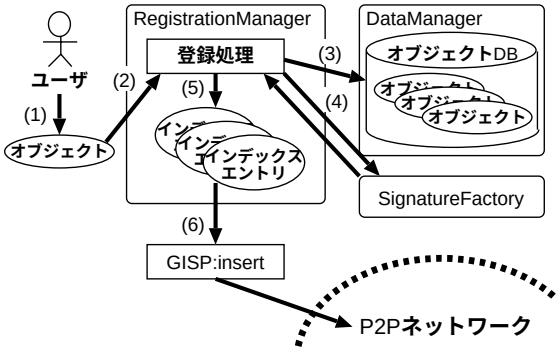


図 6　登 録 処 理

5. 設計と実装

本研究では，実計算機環境における評価検討を行うため，提案方式を採り入れたフレームワークの設計と実装を行う．本フレームワークは基本の枠組みとして JXTA と GISP を用い，これらの上に提案方式を実装したシステムを構築する (図 5).

5.1 基本となる枠組み

5.1.1 JXTA

JXTA [8] は，Sun Microsystems 社が提供する P2P アプリケーションを作成するための基本フレームワークであり，P2P プロトコルの標準化を目指し開発が行われている．JXTA は様々なプロトコルやそれに基づく API を定めている．このため開発側は提供される API を利用することで，物理的なネットワークを考慮することなく P2P アプリケーションを作成できる．

5.1.2 GISP

GISP は DHT の実装であり，基本フレームワークとして JXTA を利用する．GISP が提供する API は以下である．

- $insert$(key, オブジェクト, 制限時間);
- $query$(key, リスナー, 制限時間);

$insert$ 関数は key に基づき，オブジェクトを適当なノードへ登録する．なお GISP は明示的にオブジェクトを削除する関数を持っていない．このため，制限時間を各オブジェクトに設けることにより，これを超えたオブジェクトは自動的に削除される．一方，$query$ 関数は key に基づき，オブジェクトを検索する．問合せに合致したオブジェクトはリスナーへ逐次返される．$query$ 関数も同様に制限時間が設けられており，これを超えた問合せは自動的に削除される．GISP では検索処理を単純化するため，解を必ず返すことを保証していない．したがって，この点が DHT を用いた手法と大きく異なる．

また P2P 環境では予期しないノードのダウン等が存在する．これに対応するため，GISP は $insert$ 時および $query$ 時にオブジェクトや問合せを複製し処理を行う．一定の冗長性を持たせることにより，ある程度の耐障害性を実現する．複製数は GISP の起動時に設定することが可能である．

5.2 プロトタイプシステム

プロトタイプシステムのアーキテクチャを図 5 に示す．システム構成は，オブジェクトから特徴量を抽出しシグネチャを生成するモジュール $SignatureFactory(SF)$，登録処理を行う (インデックスエントリの生成等) モジュール $RegistrationManager(RM)$，検索処理を行う (部分問合せエントリの生成等) モジュール $QueryManager(QM)$，登録オブジェクトや検索結果を管理するモジュール $DataManager(DM)$ から成る．DM は 2 種類のデータを管理する．一つは登録要求のあったオブジェクトを管理するオブジェクト DB であり，もう一つは問合せにより得られたインデックスエントリを管理する検索結果 DB である．

プロトタイプシステムの実装には Java(Java2SDK1.4) を用いた．コードサイズは約 4000 行である．

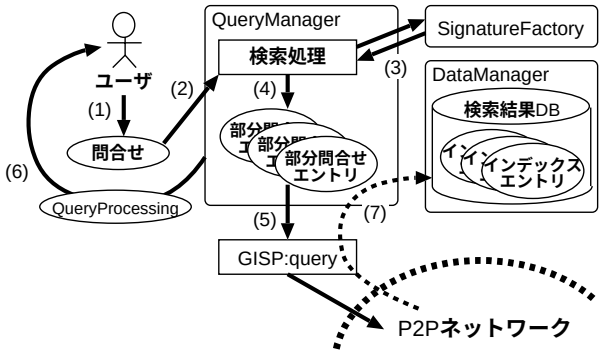
5.2.1 登録処理の流れ

本システムにおけるオブジェクト登録処理について説明する．処理の流れを図 6 に示す．まず，(1) ユーザは登録するオブジェクトを生成する．次に (2) 生成したオブジェクトを RM へ送り，(3)DM はオブジェクト DB へ追加する．(4)SF はオブジェクトを受け取り，オブジェクトシグネチャを生成し RM へ返す．さらに，(5)RM はオブジェクトシグネチャからインデックスエントリを生成し，(6)GISP の $insert$ 関数を用いて，インデックスエントリを適当なノードへ登録する．

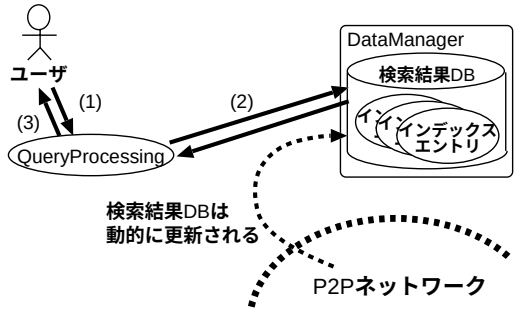
5.2.2 検索処理の流れ

本システムにおけるオブジェクト検索処理について説明する．処理の流れを図 7(a) に示す．まず，(a-1) ユーザは問合せ条件から問合せを生成する．次に (a-2) 生成した問合せを QM へ送る．(a-3)SF は問合せを受け取り，問合せシグネチャを生成し QM へ返す．さらに，(a-4)QM は問合せシグネチャから部分問合せエントリを生成し，(a-5)GISP の $query$ 関数を用いて問合せを行う．ここまでの処理が終了した段階で，(a-6)QM は $QueryProcessing(QP)$ を生成し，ユーザへ返す．また，(a-7) 問合せより得られるインデックスエントリは GISP へ非同期に送られ，検索結果 DB へ逐次追加される．

5.1.2 で述べたように GISP は問合せに対して必ず結果を返すことを保証していない．このため，本システムはすべての部分問合せエントリによる検索の終了時を知ることはできない．したがって検索結果を得る場合，ユーザが検索結果 DB へ問い合わせ，現時点で蓄積されているインデックスエントリ集合に対して解の絞り込み処理を行う．検索結果取得処理の流れは図 7(b) である．(b-1) ユーザは，まず QP に問い合わせる．



(a) P2Pネットワーク上への問合せ処理



(b) 検索結果取得処理

図 7　検 索 処 理

(b-2)QP は DM の検索結果 DB から照合対象のインデックスエントリ集合を獲得し，4.3 で述べた図 4 の (4) 以降の処理を行う．最終的に，(b-3) 絞り込み処理により得られた ANS をユーザへ返す．検索結果 DB は動的に更新されるため，ANS は常に変化する．

5.3　登録時および検索時の制限時間

GISP は DHT を用いた手法を完全に実装したものではないため，本フレームワークを実装する上で GISP との整合性を図る必要がある．特に GISP は各処理に制限時間を設定しており，これを本システムの登録処理および検索処理に組み込む．

まず登録処理の際にオブジェクトに与える制限時間について図 8 を用いて述べる．登録時の制限時間は，分散配置するインデックスエントリの利用可能な期間を意味する．まず，ユーザより登録要求のあったオブジェクト D_1 とその制限時間 t_1 はオブジェクト DB で管理される．さらに GISP を利用し，制限時間を t_1 とした各インデックスエントリ IE を P2P ネットワーク上のノードへ分散配置する．登録処理により送られたインデックスエントリは各ノードの GISP 部で管理される．GISP 部は，制限時間を超えたインデックスエントリを自動的に破棄する．このため登録から t_1 時間が経過した場合，D_1 の各インデックスエントリは破棄されてしまい検索時に利用できない．そこで各ノードは制限時間を超えたオブジェクトに対し，制限時間の更新と，インデックスエントリ集合の生成および登録処理を再度行う．

次に検索時に与えられる制限時間について図 9 を用いて説明する．検索時の制限時間は，蓄積された検索結果をキャッシュとして利用可能な期間の指定と GISP で処理される問合せの制限時間を意味する．まずユーザから生成された問合せ Q_1 は問合せ処理を行うロケータ集合 $\{LOC_A, LOC_B, LOC_C\}$ を生成する．生成された各ロケータと制限時間 t_1 は検索結果 DB で管理される．これらは GISP 部へ渡され，P2P ネットワーク上を検索する．さらに Δt 時間後 $(t_1 > \Delta t)$ にユーザが問合せ Q_2 を行った場合，先程と同様にロケータ集合 $\{LOC_B, LOC_D\}$ と制限時間 t_2 を管理する．このとき既に LOC_B は管理されているため，LOC_B の制限時間を t_2 に更新する．現時点で LOC_B の検索結果領域に蓄積されたインデックスエントリ集合は，Q_2 の検索結果を獲得するためのキャッシュとして利用することができる．さらに $(t_1 - \Delta t)$ 時間経過した場合，LOC_A および LOC_C は制限時間を超えてしまうため，蓄積された検索結果は破棄される．このように制限時間を適切に設定することで，キャッシュとしての利用度を高めることができる．

表 1　実験時の基本となるパラメタ

項目	値
key のビット長:k	12
ノード数	4
総データオブジェクト数	100
シグネチャ長:F	2^9(=512)
データオブジェクトの特徴量の数	50
問合せの特徴量の数	1
フォールスドロップ確率	0.00729

5.4　API

本システムは以下の API を提供する．GISP との整合性を図るため，各関数に制限時間を与える．

- $register$(オブジェクト, 制限時間);
- $QueryProcessing\ search$(問合せ, 制限時間);

6. 評 価 実 験

実計算機環境における評価実験を行った．本実験は，CNN のニュース記事をデータオブジェクトとして用いた．特徴量としては語 (word) を用い，オブジェクトシグネチャを生成する．また，本 API における制限時間を非常に大きく設定しておき，GISP によるオブジェクトおよび問合せの複製は行わないものとする．実験環境は，各端末を LAN によって接続したプライベートネットワークである．実験時の基本となるパラメタは表 1 である．

実験内容は，(1)GISP の基本性能評価，(2) 基本となるパラメタ時の処理時間，(3) 総データオブジェクト数を変化させた場合の処理時間，(4) ネットワーク環境の規模を変化させた場合の処理時間について測定する．本システムは GISP を基本の枠組みとして用いるため，実験 (1) によってその基本性能を評価する．さらに実験 (2) によって，本フレームワークの実処理時間を測定し詳細な検討を行う．次に実験 (3) では，データオブジェクト数を変化させた場合について，処理時間にどのような影響を与えるのかを調べる．また，現実の P2P ネットワークのノード数は非常に大きく，このような大規模なネットワーク環境を想定した場合の処理時間を検討するため実験 (4) を行う．各実験の測定時はメモリやキャッシュの内容を毎回クリア

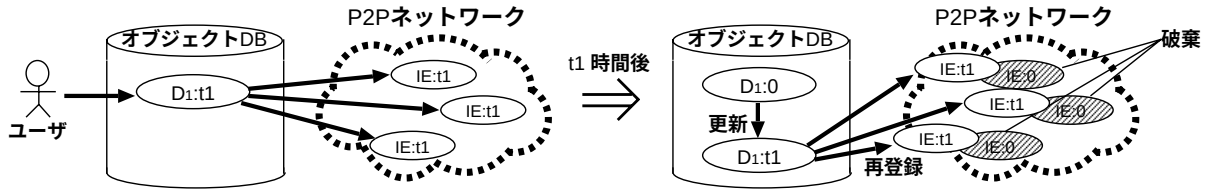


図 8　オブジェクト登録時の制限時間

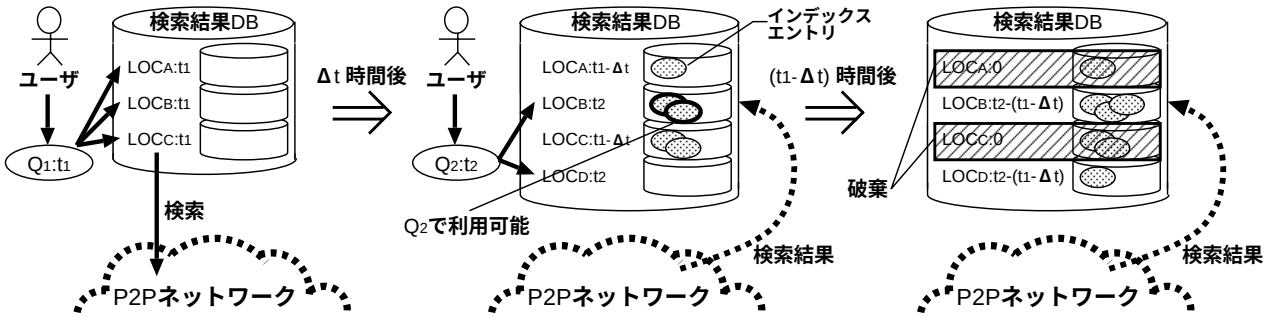


図 9　オブジェクト検索時の制限時間

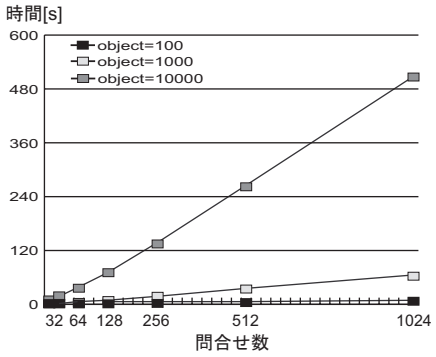


図 10　オブジェクト数を固定した場合の検索時間

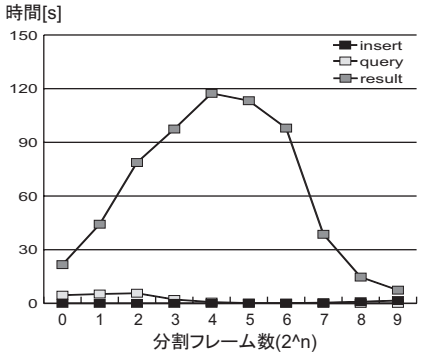


図 12　実処理時間

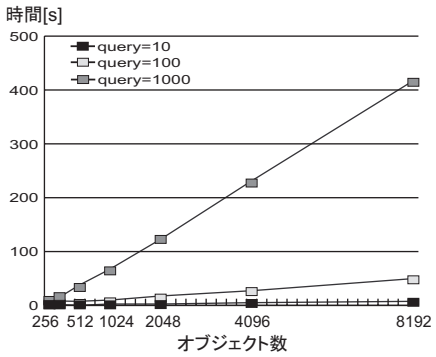


図 11　問合せ数を固定した場合の検索時間

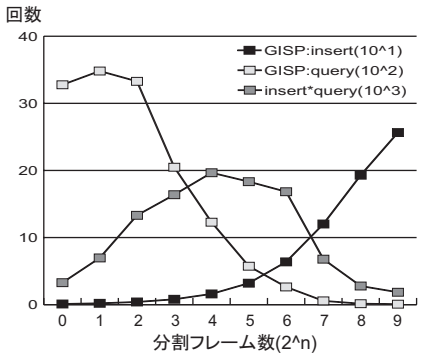


図 14　GISP API 呼び出し回数

することで，これらが処理時間に及ぼす影響を避ける．

6.1　GISP の基本性能評価

まず，GISP の基本的性能を実験により評価した．実験は，オブジェクト数を固定し問合せ数を変化させた場合と，問合せ数を固定しオブジェクト数を変化させた場合で行った．すべての問合せをほぼ同時に送り出した場合に，すべての結果を取得するまでにかかった処理時間 (ここでは単に検索時間と呼ぶ) を測定した．

実験結果は図 10，図 11 である．どちらの実験の場合も検索時間はほぼ線形的に増加していることが分かる．したがって，GISP における検索時間はオブジェクト数と問合せ数に比例することが分かった．

6.2　基本となるパラメタ時の実処理時間

表 1 のパラメタを設定した場合の評価実験を行った．実験では分割フレーム数を変化させ，3 種類の平均値を測定した．T_{insert} はあるデータオブジェクトを登録し終えるまで (GISP の $insert$ 関数の処理を終えるまで) にかかった時間であり，T_{query} はある問合せを処理し終えるまで (GISP の $query$ 関数の処理

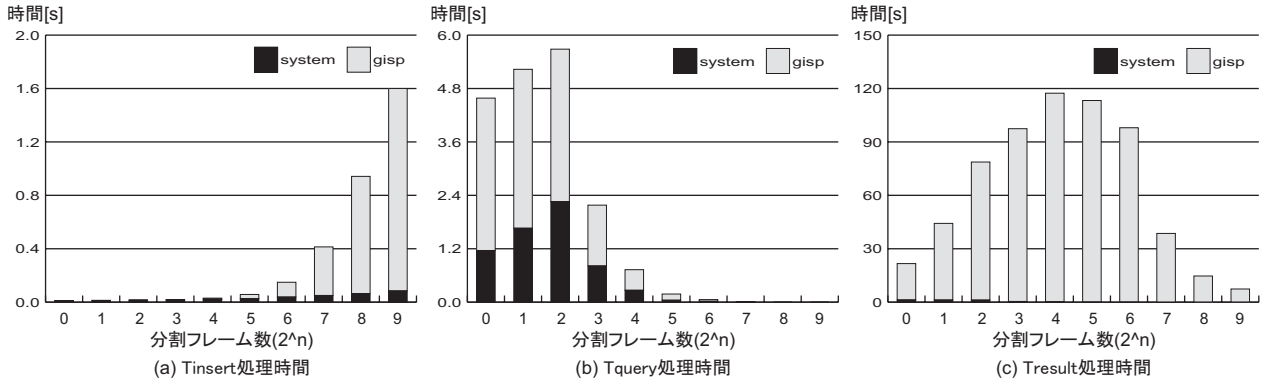


図 13　各処理時間の詳細

を終えるまで) にかかった時間である．なお本フレームワークは，すべての解を獲得するまでにかかった時間を正確に測定することはできない．しかし，実験では全データオブジェクトを事前に決定しているため，得られるべきインデックスエントリは計算しておくことができる．そこで，すべての得られるべきインデックスエントリを獲得するまでの時間と解の絞り込み処理のためにかかった時間の合計を，すべての解を獲得するまでにかかった時間 T_{result} とする．

図 12 が実験結果である．T_{insert} と T_{query} については T_{result} と比べほぼ一定の小さい値を示しており，分割フレーム数の変化による影響は比較的小さい．一方 T_{result} は，分割フレーム数により結果が大きく異なる．より詳細な検討を行うため，本システム部と GISP 部が各処理時間にどのような影響を与えているのかを測定した．これにより責任分解点を明確化する．図 13 が測定結果である．まず T_{insert} に関して，分割フレーム数が多い場合は配置するインデックスエントリ数が多くなるため，処理時間が大きくなる．一方，T_{query} に関して，分割フレーム数が多い場合は問合せを行うロケータ数が少なくなるため，処理時間が小さくなる．どちらも GISP 部の処理で時間がかかっており，本システム部の処理時間は比較的小さいことが分かる．さらに T_{result} に関しては，すべてのインデックスエントリを獲得するまでにかかった時間が大部分であり，本システム部の絞り込み処理によりかかった時間はほとんど見られない．この測定結果から T_{result} は GISP 部に大きく依存していることが分かった．そこで図 12 の評価実験における GISP の API 呼び出し回数を測定した (図 14)．分割フレーム数が少ない場合は *query* 関数の呼び出しが多く，分割フレーム数が多い場合は *insert* 関数の呼び出しが多いことが分かる．また GISP の検索時間はオブジェクト数と問合せ数に比例するため，計算量として *insert* 関数と *query* 関数の呼び出し回数の積を求めた．これは分割フレーム数が 2^4 付近で最もコストが高く，T_{result} と同様な曲線を描いている．しがたって，この計算量が T_{result} に大きな影響を与えていると考えられる．これらの実験結果から，T_{result} に関しては，分割フレーム数が 2^4 付近で最も時間がかかり，分割フレーム数が 2^0 または 2^9 付近の場合が優れている．

6.3　総データオブジェクト数を変化させた場合

総データオブジェクト数を 100，300，500 と変化させた場合

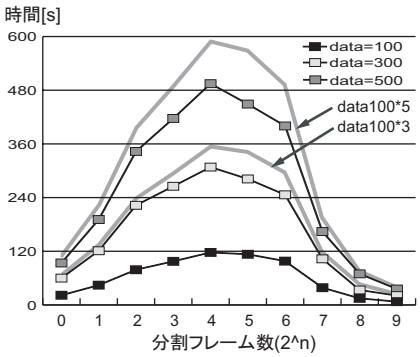


図 15　オブジェクト数を変化させた場合

の処理時間 T_{result} について計測した．一般に，データオブジェクトの増加に伴い，登録するインデックスエントリ数も均等に増加するため T_{result} は線形に大きくなることが予想される．

実験結果を図 15 に示し，さらに総データオブジェクトが 100 の場合の T_{result} を単純に 3 倍および 5 倍した値を示した．実験結果から総データオブジェクト数を増加した場合，T_{result} は増加するが，完全な線形増加にはなっていない．これは GISP の検索時間 (図 10 および図 11) が関係していると考えられる．GISP の検索時間の増加率は，オブジェクト数や問合せ数を大きくしていった場合の増加率に比べ若干小さくなっている．このことから，すべてのインデックスエントリを取得するまでの時間が短縮され，T_{result} は単純な線形増加にはならないと考えられる．

6.4　ネットワーク環境の規模を変化させた場合

本評価実験では，基本となるパラメタとして *key* の長さ k(ロケータの長さ) を 12 と定めた．k が非常に小さい場合は，同一なロケータが多く発生する可能性が高くなるため，特定のノードに登録時や検索時の負荷が集中してしまう恐れがある．このため大規模なネットワーク環境を想定した場合は，k をある程度大きくする必要がある．そこで，k を変化させた場合の処理時間 T_{result} を測定した．実験はシグネチャのフレーム分割を行わない場合 (2^0) と，最も細かく分割した場合 (ビットスライス構成 $= 2^9$) で行った．

実験結果は図 16 である．分割フレーム数が 2^0 の場合は，k を増加させると処理時間が大幅に増加する．これは k が大きく

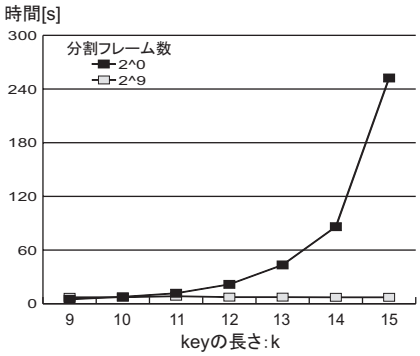


図 16　ネットワーク環境の規模を変化させた場合

なると，部分問合せエントリの検索対象ロケータ集合の要素数が大きくなり，結果的に GISP の *query* 関数呼び出し回数が非常に多くなるためである．また大規模なネットワーク環境における *query* 関数呼び出し回数の増加は，大きなネットワーク遅延を発生させる原因になり得る．一方，分割フレーム数が 2^9 の場合では，k の変化による影響はほとんど見られない．したがって，より大規模なネットワーク環境を想定した場合，ビットスライス構成の場合は少ない *query* 関数呼び出しで効率的に処理ができると考えられる．このため，ビットスライス構成の場合の方が柔軟性が高く優れていると判断できる．

7. 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，提案方式を採り入れた P2P フレームワークの設計と実装を行った．さらに実データを用いて実計算機環境上で評価実験をし，詳細な検討を行った．これにより，問合せに関してはビットスライス構成の場合が最も柔軟性が高く効率的であることがわかり，分散型シグネチャを用いた本提案方式の有効性を示すことができた．

今後の課題として，他の DHT の実装を適用した場合の比較実験や，より大規模な実計算機環境での評価実験がある．現実の P2P ネットワークのノード数は非常に多い．したがって大規模な P2P 環境下での検索処理や登録処理が，実処理時間にどのように影響するのかを調べ，適切なパラメタを設定するための評価検討が必要である．また，このような場合は各ノードが様々な問合せを行うことが考えられるため，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も行う必要がある．

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．

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] Gnutella website. http://www.gnutella.com/.
[3] Kirsten Hildrum, John D.Kubiatowicz, Satish Rao, and Ben Y.Zhao, Distributed Object Location in a Dynamic Network, SPAA'02, pp. 41-52 (2002).
[4] Daishi Kato, GISP:Global Information Sharing Protocol, IEEE P2P'02, pp. 65-72 (2002).
[5] Zheng Lin, and Christos Faloutsos, Frame-Sliced Signature Files, IEEE TKDE Vol.4, No.3, pp. 281-289 (1992).
[6] 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方式, 情報処理学会論文誌, Vol. 44, No. SIG 12(TOD 19), pp. 139-149 (2003).
[7] Ryo Matsushita, Hiroyuki Kitagawa, and Yoshiharu Ishikawa, Feature-based Distributed Object Search Using Signatures in Peer-to-Peer Environments, Proc. 19th Annual ACM Symposium on Applied Computing (SAC 2004) (2004).
[8] Project JXTA website. http://www.jxta.org/.
[9] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, A Scalable Content-Addressable Network, SIGCOMM'01, pp. 161-172 (2001).
[10] 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).