P2P 情報検索における索引とファイルの分散配置手法

倉沢　央[†]　正田　備也[††]　高須　淳宏[††]　安達　淳[††]

† 東京大学大学院 東京都千代田区一ツ橋 2-1-2
†† 国立情報学研究所 東京都千代田区一ツ橋 2-1-2
E-mail: †{kurasawa,masada,takasu,adachi}@nii.ac.jp

あらまし　ピア・ツー・ピア (P2P) ネットワークを用いた情報検索では，低コストでありながら負荷分散や高いスケーラビリティが簡単に実現可能である．従来のノード単位でキーワードのインデキシングを行う手法では，ノードの評価が影響するため検索漏れを引き起こしやすい．また，同一ファイルの区別をしにくいためファイルの冗長化が難しい．そこで本稿では，P2P 情報検索における索引とファイルの分散配置手法，Concordia を提案する．(k, n) 閾値法を用いてファイルを分散符号化し，文書におけるキーワードの重みに応じてキーワードに対応付けする分散情報の数を決め，DHT 上にインデックスと分散情報を統合して配置することで，ファイルのクエリとの適合度を考慮した検索と，ファイルの総量を抑えた負荷分散とノードの離脱への対策を備えた効率の良い冗長化を実現する．

A Distributed Index and Data Allocation Scheme for P2P Information Retrieval

Hisashi KURASAWA[†], Tomonari MASADA[††], Atsuhiro TAKASU[††], and Jun ADACHI[††]

† The University of Tokyo 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, JAPAN
†† National Institute of Informatics 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, JAPAN
E-mail: †{kurasawa,masada,takasu,adachi}@nii.ac.jp

Abstract　Many Peer-to-Peer information retrieval systems use keyword-peer index and require peer selection techniques. Peer selection tends to fail the most relevant file and cannot identify replica files. We propose Concordia,a new distributed index and data allocation scheme for P2P information retireval, that searches and gathers relevant files based on its relevance to the query and realizes efficient redundancy for load balance and node departure. Our system makes n pieces from a data with (k, n) threshold scheme and places pieces based on the weight of a keyword on the peer related to the keyword index in DHT.

1. はじめに

近年，多くのコンピューティング資源がネットワークによって結ばれ，爆発的な勢いで情報が増大し，流通するようになった [11]．それら大量の情報を効率良く扱え，最適な情報を得られるものとして，検索エンジンが大きな役割を担っている．現在主流の検索システムは，Google [1] に代表される集中型である．クローリングを頻繁に行い，インデックスを更新していくことで検索対象を増大させている．ある程度中規模以上の検索システムの構築，運用には，負荷分散やスケーラビリティなどについての専門的な知識と，高いコストが必要となることが知られている．これに対して，低コストでありながら負荷分散や高いスケーラビリティを簡単に実現できるものとして，P2P ネットワークを用いた検索システムが近年注目をあびている．

これまで P2P ネットワークを使った様々な情報検索手法が提案されているが，既存研究の多くは共有目的のファイルごとにインデキシングをせず，ノードの持つすべての文書の概要をもとにノードごとにキーワードのインデキシングをしている [7] [4] [17]．この方式では，ノードのもつファイルコレクションの評価をもとにファイルの収集を行うため，個々のファイルごとにクエリとの適合度を評価することが難しく，検索漏れを引き起こしやすい．また，既存研究ではノードの離脱や負荷分散を想定した共有目的のファイルの冗長化とファイル配置への検討が少ない．仮にファイルの複製を配置しても，既存の手法ではノード単位のイ

ンデックスでは同一ファイルを検索の段階では区別できないため，複製ファイルを持つノード宛に無駄なクエリを投げてしまう．

それに対して，筆者らはインデックスの作成とデータの配置，検索，復元の 3 ステップで構成される，P2P 情報検索における索引とファイルの分散配置手法，Concordia を提案する．Concordia では，共有目的のファイルごとにファイルにおけるキーワードの重みを参照可能なインデックスを用いる．それにより，同一ファイルを同定することを可能とするだけでなく，ファイルごとにクエリへの適合率を意識した検索を可能とする．また，Concordia では，共有目的のファイルに対して (k, n) 閾値法を用いて分散符号化をし，それをキーワードの重みに基づいて配置する．それにより，インデックスとファイルの配置を統合的に扱うことができ，ファイルの総量を抑えながらも負荷分散とノードの離脱への対策を備えた効率の良い冗長化を実現する．

本稿では，まず第 2 章で既存の P2P 情報検索手法の問題点を明らかにする．第 2 章で述べた問題に対し，第 3 章にて P2P 情報検索における索引とファイルの分散配置手法について述べる．第 4 章で本手法の評価を行い，第 5 章で考察を行う．

2. P2P 情報検索の関連研究

2.1 P2P ネットワークを使った情報検索

P2P ネットワークとは，不特定多数のサーバが対等な立場で作る分散システムのネットワークを言う．各ノードはアプリケーションレベルで相互に接続し，TCP/IP 層の上に仮想的なネットワークを形成している．アーキテクチャは大きく 2 つの方式があり，一つは Hybrid P2P 型，もう一つは Pure P2P 型である．本研究では Pure P2P 型を扱う．

Pure P2P 型は純粋にピアのみでネットワークを構築する形態である．ネットワークに参加しているピアのうち一つとまず接続し，そのピアを介して他のピアのアドレスを集めることを繰り返すことでネットワークを形成していく．Hybrid P2P 型と比べてノード数やファイル数について高いスケーラビリティをもつ．また，中心となるサーバを持たないため，管理や認証は組み込みにくいが，単一障害点を持たないため障害に強い．

P2P 情報検索では，そのような全体の管理が困難なネットワークにおいて，共有目的のファイルを効率よく発見・収集するため，集中型の検索システムと同じくインデックスの作成を行う．そして，ファイルと同様にインデックスも効率よく発見されるように工夫した配置を行う．集中型の検索システムでは一般的に再現率，適合率という 2 つの評価尺度が用いられるが，それらを P2P 型で高めるときには以下の課題が存在する．再現率を高めようとすると，ノードへ検索クエリをなるべく多く拡散させることとなり，トラフィックが多くなってしまう．一方，適合率を高めようとすると，他の文書よりも適合しているかを正しく判断する必要があり，全体を把握しにくい P2P ではその判断が難しい．P2P 情報検索の既存手法はそれらの問題を踏まえた設計となっている．

P2P 型を使う利点として，負荷分散や高いスケーラビリティを低コストで実現できるという点が挙げられる．一般的に集中型の検索システムは，規模が大きくなるにつれシステムコストが高まり，運営・構築が困難になる．P2P 型は，資本力のない巨大なコミュニティでの情報共有や特定の代表者のいないコミュニティで有効であると考える．

一方，P2P 型の欠点として，ネットワーク全体を把握するのが難しいこと，大量のトラフィックを引き起こしやすいこと，ノードの参加・離脱が起きうること，検索応答がピアのネットワーク環境に大きく依存することが挙げられる．これら欠点を補うため，DHT を用いてハッシュ空間で構造的にノードとファイルを管理する [16]，Bloom Filter [6] を用いて通信量を軽減する，レプリケーション [10] や分散符号化 [9] して扱う共有目的のファイルを冗長にするなど，様々な対策が考案されている．

2.2 既存のアプローチ

Rutgers University で研究されている PlanetP [7] では，キーワードとノードのアドレスを対応づける Bloom Filter [6] を用いたインデックスをグローバルに同期して保持することで文書検索を可能にしている．インデックスの同期には，フラッディングを繰り返して全ノードへ情報を行き渡らせる Gossip アルゴリズムが用いられている．文書を検索するときは，まず同期して保持しているグローバルのインデックスから探したいキーワードを含む文書を持つノードを探し，絞り込んだノードに対してクエリを投げることで適切な文書を取得する．トラフィックを軽減するために，クエリに対して評価の高いノードから接続して文書を収集する設計になっている．ノードの評価には，IPF というクエリに含まれるキーワードを保持するノード数の逆数を利用している．Bloom Filter では，接続するノードを絞る段階で，各文書におけるキーワードの局所的な重みは考慮されない．

Max-Planck-Institut fuer Informatik で研究されている MINERVA [4] では，各ノードが担当するキーワードを DHT におけるハッシュ値を元に割り振り，文書の整理を行う．共有目的のファイルをもつノードは，ローカルに保持する文書コレクションに含まれるキーワードをまとめた概要を作成する．そして，共有目的のファイルを持つノードは，その概要に含まれるキーワードを担当するノードのインデックスに自分のアドレスを登録する．その結果，各ノードは，担当するキーワードを保持するノードのアドレスを把握することができるので，それを PeerList にまとめて保持する．文書を検索するときは，クエリに含まれるキーワードを担当するノードに接続することで，容易にキーワードに関連する文書を持つノードを見つけることができる．トラフィックを軽減するために，CORI や cdf-ctf といった手法を用いることで，ノードの持つ文書におけるキーワードの占める割合を考慮した評価を算出し，接続するノードをさらに絞り込む設計になっている．ノードの選択手法として，各ノードの持つ文書コレクションの重複度合いを考慮して，収集するファイルの再現率を高める研究もされている [5]．

いずれの研究も，ノードを単位としてローカルに保持す

表 1　既存研究との比較

	indexing	data allocation	replication or encoding	peer selection
our scheme	DHT	the nodes related to data index	(k,n) threshold scheme	no need
PlanetP	Gossip	publisher	-	IPF
MINERVA	DHT	publisher	-	CORI2,cdf-ctf,and so on

る文書中に出現するキーワードのインデックスを作成している．なぜなら，既存研究では，共有目的のファイルごとに出現するキーワードのインデックスを作成することはスケーラビリティに問題があるからである [5]．しかし，ノードごとのインデックスで検索を行うと，ノードの評価が必要となるうえ，個々のファイルごとのクエリとの適合度の評価を容易に比較することができず，検索漏れを引き起こしやすい．また，既存研究ではノードの離脱や負荷分散を想定した共有目的のファイルの冗長化とファイル配置への検討が少ない．ノードごとのインデックスでは同一のファイルを同定が難しいため，複製ファイルを保持するノードへ無駄にクエリを投げてしまう以上を踏まえて，ファイルごとにクエリとの適合度を考慮した検索・収集が可能で，さらに効率の良い冗長化と配置を行う特徴を持つ，P2P 情報検索における索引とファイルの分散配置手法が必要であると考える．関連研究と本手法を比較したものが表 1 である．

3. Concordia のアーキテクチャ

Concordia の機能的な特徴は次のようにまとめられる．

- ファイルごとにキーワードの重みをインデックスに登録することでクエリとの適合度を取得可能
- (k,n) 閾値法によるノードの離脱や負荷分散を想定したファイルの冗長化
- キーワードの重みに基づいた分散情報の配分にすることでクエリに適合するファイルの収集効率の向上

3.1 Concordia の概要

Concordia は，インデックスの作成とデータの配置，検索，復元の 3 ステップで構成される．

インデックスの作成とデータの配置では，図 1 のように，まず，共有の対象とするファイルからキーワードの抽出とファイルの (k,n) 閾値法を用いた分散符号化を並行して行う．次に，得られたキーワードのうち，ファイルにおける重みの大きいキーワードほど多くのファイルの分散情報が割り当てられるように配分する．キーワードごとに配分された分散情報とファイルの復元や収集に必要な情報をパッケージ化し散布ファイルとしてまとめ，DHT 上のキーワードのハッシュ値に対応するノード i 個宛てに送信する．これにより，ファイルごとにそのファイルにおけるキーワードの重みが取得可能となる．また，ノード i 個に対して同一の散布ファイルの配置，(k,n) 閾値法によるファイルの冗長化，ファイルにおけるキーワードの重みに基づいた配置により，ファイルの総量を抑えながらも負荷分散とノードの離脱への対策を備えた冗長化を行っている．

検索では，図 2 のように，まず，ユーザが作成したクエリからキーワードを抽出する．キーワードごとにハッシュ値を計算し，DHT で該当するノードに接続し，そのノードが保持する散布ファイルを収集していく．つまり Concordia では，散布ファイルはインデックスを保持するノードから取得する．検索を行うことで，クエリとの適合度に相関した量のファイルの分散情報がローカルに保存されることになる．

復元では，分散情報の数が閾値を上回っている場合，共有目的のファイルの復元することができる．復元したいファイルの分散情報が不足している場合は，ユーザが既に持っている散布ファイルから収集に必要な情報を参照し，ユーザは必要な量だけ新たに散布ファイルを収集する．

本研究で用いる語句を以下にまとめる．

共有目的のファイル　P2P 情報検索で検索及び共有の対象となるファイルを意味する．

分散情報　共有目的のファイルを (k,n) 閾値法を用いて分散符号化した n 個のファイルを意味する．n 個のうち k 個の分散情報から共有目的のファイルを復元することができる．

散布ファイル　キーワードとファイルにおけるキーワードの重み，重みに対応した量の分散情報，共有目的のファイルの復元に必要な情報を詰めたファイルを意味する．キーワードのハッシュ値に該当するノードが保持するファイルである．

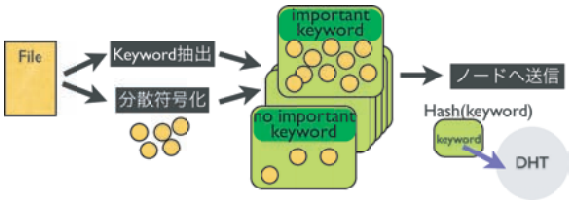

図 1　インデックスの作成とデータの配置

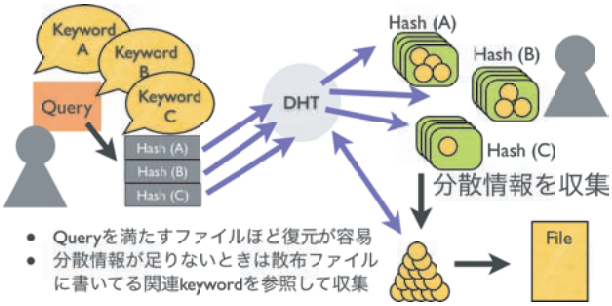

図 2　検索と復元

3.2 インデックスの作成とデータの配置

3.2.1 P2P 環境でのキーワード抽出と重み付け

本手法をより効率の良いものにするうえで，キーワード抽出と重み付けは最も重要な要素である．ここでは文書検

索を例に，一般的な情報検索アルゴリズムを P2P 環境に適用した計算法を述べる．

情報検索の代表的な手法に，ベクトル空間モデル (Vector Space Model) と確率モデル (Probabilistic Model) がある．ベクトル空間モデルは，文書とクエリを多次元のベクトルという中間的な媒体で表現し，ベクトル間の類似度を計算することにより，文章の類似検索を実現する手法である．一方，確率モデルは，索引語の文書における出現確率を用いた手法である．

本研究では，キーワードの重み付けに Okapi(BM25) [14] と呼ばれる確率モデルの式を元にした以下の式を用いた [8]．

$$w(t) = \log \frac{N}{df} \cdot \frac{(k+1) \cdot tf}{k\{(1-\alpha) + \alpha \cdot \frac{dl}{avdl}\} + tf}$$

w はキーワードの重み，N は文書の総数，tf は文書 d におけるキーワード t の出現頻度，df は全文書におけるキーワード t を含む文書数，dl は文書 d の長さ，$avdl$ は文書の長さの平均値，k，α は定数を表す．クエリとの適合度 R_Q は，クエリに含まれるキーワードそれぞれについて上の式を計算した和

$$R_Q = \sum_{t \in Q} w(t) \tag{1}$$

を用いる．

P2P 環境にて上の式を適用する場合，キーワードの局所的重みである tf はローカルの情報のみで計算できるが，キーワードの大域的重みである df は他ノードのもつ情報を手に入れないと計算できない．我々は，既存研究 [4] [17] と同じように DHT を用いて構造的にキーワードのインデックスを作成し，DHT 上に配置された情報を元にキーワードの大域的重みの値を得ることにした．本研究では先に述べたように，各ノードは自分の担当するハッシュ値に該当するキーワードに対応する散布ファイルを管理する設計である．個々のキーワードのディレクトリには，共有するファイルごとに散布ファイルが置かれることになる．これを利用することにより，任意のキーワードのハッシュ値を担当するノードにアクセスすることで，そのキーワードを含む文書の数を知ることができる．トラフィックを軽減するために，他のノードに接続する機会があるたびにハッシュ値とファイル数をまとめた辞書を交換しあうことでローカルの辞書を更新していくようにし，必要に応じてノードに接続するという設計にした．

文書の総数 N と文書の長さの平均値 $avdl$ の値の算出は，各ノードの保持する文書数が動的に変化するため難しいと思われる．そこで，本研究では簡略化し，あらかじめ P2P ネットワークの規模と流通ファイル数を考えて，プログラム配布者が定数を設定することにした．

3.2.2 ファイルの分散符号化手法

ファイルの分割には (k, n) 閾値法を用いる．共有目的のファイルを n 個の分散情報に分散符号化し，それらの任意の k 個から共有目的のファイルを復元できる手法である．つまり，n 個のうち $n-k$ 個の分散情報が収集できなかったり，分散情報を保持している一部のノードがネットワークに参加していなくても共有目的のファイルの復元が可能である．(k, n) 閾値法は $k-1$ 次関数上の n 個の座標を用いたもの [15] や，k 個の文字列を使った連立方程式を用いたもの [9] がある．いずれもファイルの復元は k 個の連立方程式の解を求める計算となる．

(k,L,n) ランプ型秘密分散法 [18] は，分散対象の情報を $L+1$ 個のビット連結として扱い，それを $k-1$ 次関数の係数とする手法である．式で表すと以下のようになる．

$$S = S_0||S_1||S_2||\cdots||S_L$$

$$y = S_0 + S_1 x + \cdots + S_L x^L + r_1 x^{L+1} + \cdots + r_{n-L} x^{k-1} \quad mod\ p$$

S は共有目的のファイル，|| はビット連結演算子，p は $p > max(S_i)(0 \leqq i \leqq L)$ を満たす素数を示す．分散符号化された分散情報は，この関数上の (x, y) の座標情報 n 個となる．共有目的のファイルサイズを S とすると，個々の分散情報のサイズ m は $m \geqq \frac{S}{L}$ となる．

本研究ではこの (k,L,n) ランプ型秘密分散法を用いた．L の値を変化させることで共有目的のファイルの分散符号化のセキュアな度合いを決めることができるが，今回は分散符号化することが目的なので，簡単のため $L-1=k$ とした．

3.2.3 分散情報の配置

ここでは，分散情報の配置手法について述べる．P2P 情報検索では，クエリとの適合度の高いファイルを探せるだけでなく，クエリに適合するファイルの収集が容易であることが望ましい．検索クエリに対応するインデックスを保持するノードが，クエリとの適合度の高いファイルを保持していると，ユーザが検索過程で拡散するクエリ数が少なくすみ，検索と収集の効率が良いと言える．クエリを投げたときにアクセスする可能性の高いノードすべてにファイルの複製を配置すると，ネットワーク全体で扱うファイルの総量が大きくなってしまう．そこで，本研究ではファイルの複製の代わりに 3.2.2 で述べた分散情報を配置することにした．

ファイルのクエリとの適合度は，式 (1) で述べたように，クエリに含まれるキーワードのファイルにおける重みの和を用いる．それゆえ，ファイルにおけるキーワードの重みに基づいて，分散情報をキーワードの散布ファイルを保持するノードに配置することで，分散情報の収集がクエリとの適合度の高いファイルほど容易になる．

本研究では以上の理由から，ファイルにおける重みの大きいキーワードほど多くのファイルの分散情報が割り当てられるように配分する．キーワードごとに配分された分散情報とファイルの復元や収集に必要な情報をパッケージ化し散布ファイルとしてまとめる．

3.2.4 DHT を用いたノードとファイルの管理

DHT(Distributed Hash Table) ではノードのアドレスとファイルが共通のハッシュによって管理される．ノードは，自分のアドレスとハッシュ値が近いファイルを保持する．ハッシュを使うことにより，データを均一に分散すること

ができ，スケーラビリティを高め，ファイルの存在の有無を効率的に判断できる．検索手法は DHT によって多少異なる．代表的な例として，円構造の Chord [16]，木構造の Tapestry [19]，多次元空間の CAN [13] などが挙げられる．

本研究ではノードの管理が他の DHT よりも比較的容易な Kademlia [12] を参考に DHT を作成した．Kademlia では，直線構造のハッシュ空間を形成する．距離の定義はハッシュ値 2 つの XOR の計算値で 1 のでるビットの位置で決まる．Kademlia では，クエリとして投げたハッシュ値に近い k 個のノードのアドレスを $O(\log N)$ で取得できる．ノードは DHT 上に配置され，担当するハッシュ値が割り振られる．そして，ノードはハッシュ値に対応するキーワードに関連する散布ファイルを受け取り，保持することになる．

3.2.5 散布ファイルの管理

散布ファイルは，分散情報と復元や収集に必要な情報をパッケージ化したものであり，キーワードの数だけ作成される．

ここではまず，散布ファイルの構造について述べる．散布ファイルには，キーワードとそれに対応する分散情報だけでなく，ファイルの収集・復元に必要と思われる以下の情報を含める．

- ファイルにおけるキーワードの重み
- 共有目的のファイルのファイル名
- 共有目的のファイルのチェックサム
- 散布ファイルの作成時刻
- 共有目的のファイルのスニペット
- 共有目的のファイルの関連キーワード
- 散布ファイルの総数
- 分散情報の総数と閾値

ファイルにおけるキーワードの重みは，3.2.1 にて示したキーワードの重み付けの計算で得られた値である．散布ファイルの作成時刻は，散布から一定時間過ぎたファイルを削除するときに参照する．共有目的のファイルのスニペットは，復元するファイルを選択するときに参考とする情報である．共有目的のファイルの関連キーワードには，分散情報を多く含むキーワードを一定量含む，ランダムに選択したキーワードを想定している．分散情報が不足する場合にこれを参照して他の散布ファイルを収集する．

次に，散布ファイルの配布について述べる．共有目的のファイルを保持するノードは，時間 T_{send} ごとに分散情報を詰めた散布ファイルを，キーワードのハッシュ値に近いノード i 個に送信する．送信した際に，ハッシュ値と散布ファイルの数の対応関係を取得し，散布ファイル作成時のキーワードの重み付けに利用する．仮に散布ファイルが既に異なるノードによって配置済みであった場合は特に何も送信せず，時間 T_{wait} 後に再度アクセスする．これにより，共有目的のファイルの分散情報を関連ノードにインデックスと統合して配置することができ，また，同一ファイルの無駄な冗長化を避けることができる．

最後に，散布ファイルの更新について述べる．すべてのノードは，自分のハッシュ値に該当するキーワードの散布ファイルを保持する．時間 T_{check} ごとに，保持する散布ファイルについて，ハッシュ値の最も近いノードがないか確認を行い，自分のノードよりも最適なノードを見つけ，かつそのノードが該当する散布ファイルを保持していない場合は転送を行う．保持する散布ファイルのうちその散布ファイルの作成時刻から時間 T_{delete} 経過したものは削除する．

ノードがネットワークに参加する時は，DHT に参加する処理以外は特になにも行わない．ネットワークからの離脱時は，あらかじめ離脱が予想できる場合は，自分のノードに近いハッシュ値のノードに保持する散布ファイルをすべて転送する．突如の離脱の場合は転送は不可能であるので諦める．

3.3 ファイルの検索

検索ではまず，ユーザの作成したクエリから，複数のキーワードを抽出する．そして，それぞれのキーワードについて，インデックスを保持するノードに DHT を用いて接続する．インデックスを保持するノードでは，少しでもクエリに適合する共有目的のファイルすべてについて，ファイルにおけるキーワードの重みに相関した量の分散情報を散布ファイルとしてまとめて保持している．ユーザの要望する適合率，再現率を満たすようにそれら散布ファイルを収集する．つまり，適合率の高いファイルを要望する場合はファイルにおけるキーワードの重みの和が一定以上を満たす散布ファイルを，再現率を高めたいときはノードの保持する散布ファイルをなるべく多く収集する．このようにして，ユーザがクエリを投げて散布ファイルを収集することにより，クエリへの適合度に相関した量の分散情報をローカルに蓄えていく．

3.4 分散情報からの復元

復元では，まずローカルに保持している収集した散布ファイルを共有目的のファイルのチェックサムに基づいて整理し，その中で分散情報が閾値を上回っているファイルについてデコードする．復元されたファイルの中に探しているものがない場合は，足りない分散情報を新たに収集する．適合度の高いファイルについて収集するか，散布ファイルに含まれるスニペットを参照してファイルを選択して収集するかは，ユーザが決める．新たに収集する際には，散布ファイルの中の関連キーワードを参照して，復元したいファイルの散布ファイルの置かれているハッシュ値を得る．

4. 評　　価

Concordia は，既存研究と比べて，時間をかけて詳細なインデックスを作成することで，ファイルにおけるキーワードの重みをファイルごとにインデックスから取得可能にすることを目指している．また，(k, n) 閾値法による冗長化とファイルにおけるキーワードの重みを踏まえた配置により，ファイルの総量を抑えながらも負荷分散とノードの離脱への対策を備えた冗長化を備えていることが特徴である．

4.1 既存手法との比較

4.1.1 インデックスの作成時間と登録数

ここでは，ノードが作成するインデックスについて考える．既存手法では，各ノードは，ノードの保持する文書コ

レクションに出現するキーワードをインデックスに登録する．つまり，ローカルに保持する文書をまとめて扱う．インデックスの作成時には，文書コレクション全体でのキーワードの出現頻度やキーワードを含むファイル数が評価に用いられる．一方，Concordia では，ファイル単位でインデックスに登録する．つまり，ローカルに保持する文書は独立して扱う．

ノードのもつ文書数を N_{doc}，文書 D_i に含まれるキーワード集合を C_{D_i}，そのキーワード数を N_{D_i}，文書コレクションに含まれるキーワード集合を C_{all}，そのキーワードの数を N_{all}，キーワード t についての重みの計算に必要な時間を $f(t)$ とすると，ノード単位でのインデックス作成に必要な時間 T_1 は，

$$T_1 = \sum_{t \in C_{all}} f(t)$$

となり，登録数は，N_{all} となる．

一方，Concordia でインデックス作成に必要な時間 T_2 は，

$$T_2 = \sum_{i=1}^{N_{doc}} \left(\sum_{t \in C_{D_i}} f(t) \right)$$

となり，登録数は，$\sum_{i=1}^{N_{doc}} N_{D_i}$ となる．

インデックスの作成では，Concordia は既存手法と比較して，保持するファイル間のキーワードが重複しているとき，その重複分のキーワードのインデックス作成時間が余計にかかることがわかった．また，登録数についても，Concordia は既存手法と比較して，保持するファイル間のキーワードの重複が多いほど時間がかかることがわかった．

4.1.2 複製ファイルの取り扱い

ここでは，同一の共有目的のファイルを複数のノードで保持している状況を考える．

ノード単位でのインデックスを作成する手法では，個々のファイルを区別してインデックスに登録することが不可能である．それゆえ，複製ファイルを持つノードが増えると，ファイルの検索の際にそれらのノード宛にクエリを投げてしまい無駄なトラフィックを引き起こしてしまう．また，ファイルの複製を持つノードが多いと，検索結果において，複製の少ないファイルを持つノードを発見することが困難になってしまう．これを悪用すると，特定のファイルを意図的に検索で発見しやすくすることができてしまう．

一方，ファイル単位でインデックスを作成している Concordia では，複製ファイルが重複してインデックスに登録されることはない．設計上，検索結果として得られる結果とネットワークで共有される分散情報は，複製ファイルを持つノード数に依存しない．複製ファイルを持つノードが多くても，インデックスの更新が速やかに行われるという程度しか影響を持たない．

複製ファイルについては，本手法は重複して扱うことがなく，無駄なトラフィックを生み出しにくい設計であると言える．

4.1.3 (k, n) 閾値法を用いる有用性

本手法では，ファイルサイズの総量を抑えながらも共有目的のファイルに関連したインデックスを保持する複数のノードにファイルを配置する目的で、(k, n) 閾値法を用いて共有目的のファイルを冗長化している．ここではファイル単位でのインデックスを本手法と同様にして単純に複数ノードにファイルの複製を配置する場合と比較し，(k, n) 閾値法による冗長化の有用性について考える．ファイルサイズの総量，負荷分散，ファイルの損失確率，ファイルの復元に必要な計算量，そしてそれ以外の分散符号化の利点という 5 つの点について考察する．

a） ファイルサイズの総量

まず，ファイルサイズの総量について考える．共有目的のファイルのファイルサイズを F とおくと，分散情報 1 つのファイルサイズはおよそ $\frac{F}{k}$ となり，全体では元のファイルサイズの $\frac{n}{k}$ 倍の容量である $\frac{F \cdot n}{k}$ となる．ファイルへのニーズにかかわらず，数倍のファイル容量の増加が必要となる．人気に基づいた冗長という点では，単純に複製を配置する手法の方が柔軟に対応できると言える．

b） 負 荷 分 散

共有目的のファイルのサイズが大きいときは，単純に複製を配置すると，少なからずノードによって保持するファイルサイズにばらつきが起きる．本手法ではファイルにおけるキーワードの重みに基づいてファイルの配分を行うので，分散情報の一極集中を避け，保持するファイルサイズのばらつきを減らす設計となっている．本手法では，閾値を満たす分散情報でファイルを復元できるため，ファイル収集のアクセスが一極集中することもなく，効率の良い負荷分散が可能となっている．

c） ファイルの損失確率

ファイルの損失確率については，P2P 情報検索システムは，ノードの突如の参加・離脱が起きてもクエリに適合するファイルが常に入手可能な設計が望ましい．単純に共有目的のファイルを複製したものを異なるノードに配置した場合と，(k, n) 閾値法を用いた本手法とで，ネットワーク全体でのファイルサイズの総量が等しい状況でのファイルの損失確率の比較を行ったのが表 2 である．p はノードの離脱率，F はファイルサイズを表す．仮に，$p = 0.8$，$n = 100$，$k = 10$ とすると，損失確率は通常のファイル複製の場合は 10.7%であり，一方，(k, n) 閾値法を用いた場合は 0.233%〜10.7%となる．(k, n) 閾値法を用いた本手法では，分散情報を保持しているノードの数が閾値よりも多ければファイルの復元が可能である．それゆえ，ファイルの損失に柔軟に対応することができ，同じファイルサイズでは単純に複製を行うよりも効率の良い冗長化が可能であると言える．分散情報を保持するノードが多ければ多いほどファイルを復元しやすくなるが，それと同時にファイルの収集時に接続するノード数も増加してしまうため，トレードオフの関係となる．

d） 分散符号化を用いることで必要な計算量

(k, n) 閾値法による分散符号化を用いたときの計算量について考える．(k, n) 閾値法での分散符号化も復号化も，必要な時間は個々の手法によって多少異なるが，およそ $O(n)$ で可能である．分散情報からのファイルの復元はいずれも連立一次方程式を解くことに帰着するものが多い．連立一

表 2　情報複製手法についての比較

	replica	our scheme
num of nodes holding data	$\frac{n}{k}$	$\frac{n}{k} \sim n$
file size (piece)	F	$\frac{F}{k}$
file size (total)	$\frac{F \cdot n}{k}$	$\frac{F \cdot n}{k}$
lost probability	$p^{\frac{n}{k}}$	$p^{\frac{n}{k}} \sim \sum_{i=0}^{k-1} {}_nC_i p^{n-i} \cdot (1-p)^i$

次方程式の解法は，ガウス・ジョルダン法では $O(n^3)$ の計算負荷となる．LU 分解では $O(\frac{n^3}{3})$，同じ係数行列で右辺ベクトルだけ異なるものを続けて計算する場合は $O(\frac{n^2}{2})$ となる．いずれの解法でも，ファイルサイズが大きくなるにつれ，ファイルの復号にかかる時間は大きく増加することがわかる．単純に複製を配置する手法ではデコードの時間はそもそも必要ないため，ファイルサイズの影響を受けないで済む．

e)　(k, n) 閾値法を用いるその他の利点

最後に (k, n) 閾値法を用いる上記以外の利点について考える．n 個の分散情報のうち k 個を集めることでファイルの復元を可能とする (k, n) 閾値法であるが，k 個に満たない個々の分散情報のみでは復元は困難である．本手法で用いる (k,L,n) ランプ型秘密分散法では，L の値を変化させることで，復元の困難さを変化させることが可能である．つまり，各ノードが保持する散布ファイルに含まれる分散情報を k 個に満たない数にすることで，ノードのもつ情報のみでは共有目的のファイルの復元を難しくすることができる．これを利用することで，例えば，秘密情報を検索可能な状態で共有するときに本手法を利用した場合，ノードの盗難にあってもそのノードがネットワークに参加できないようにすばやく対策がとられれば，秘密情報の流出の危険性を幾分か下げることができる．

4.1.4　Concordia の特性

以上を踏まえて，Concordia が得意とする状況と不得意とする状況をまとめる．

Concordia が得意とする場面として，ノードの参加離脱が頻繁な状況がまず挙げられる．(k, n) 閾値法による効率的な冗長性により，単純に複製を作るよりも共有目的のファイルを復元することを容易にしている．また，ノードのもつ情報が偏っていない状況において，ファイル単位のインデックスを用いる Concordia は有効である．ノードの評価が加わらないことで検索漏れを防ぐことができる．他には，複製ファイルが大量に存在する状況，複製の少ない特定のファイルへのニーズが一極集中する状況が挙げられる．

一方，不得意とする場面として，計算資源の乏しい状況が挙げられる．Concordia は，インデックスの作成，ファイルの復元が既存手法に比べて時間も計算コストも必要であることから，豊富な計算資源を持つ環境でないと運用が難しい．また，人気に基づいた冗長性の調整をしたい場合，本手法は不利である．他には，ネットワークに参加しているノード数が少ないときは，ノードが保持する散布ファイルが膨大になってしまうので，既存のピアの評価を用いる手法の方が有効であると思われる．

4.2　ファイル復元に必要なキーワード数

Concordia は，(k, n) 閾値法による冗長化を行っている．その利点と欠点は e) で述べたとおりである．分散符号化により分散情報を生成し，3.2.3 で述べたとおり，Concordia ではファイルにおいて重みの大きいキーワードほど分散情報を多く配分している．この配分方法のとき，どのくらいのキーワード数で閾値を満たす分散情報を収集できるかについてここでは実際の文書情報を用いながら考える．

どのくらいのキーワード数で閾値を満たす分散情報を収集できるかという議論は，キーワード数個のファイルにおける重みの和が，共有目的のファイルから得られるキーワードの重みの総和の一定割合 $\frac{k}{n}$ を満たすか否かという議論と等しい．そこで，文書におけるキーワードの重みの総和の一定割合を占めるのに最低限必要なキーワード数について実験を行った．

実験に使う文書情報として NTCIR-4 PATENT [3] の一部である 2002 年度前期の日本国特許公報全文，計 186,184 文書を用いた．形態素解析には MeCab [2] を使い，キーワードの重み付けには 3.3 の式を用いた．式に含まれるパラメータは α を 0.5，k を 40 とした．平均文書長は 10,531 であった．本実験はキーワードの重みの分布を調べる目的であるので，検索精度を向上させるためのテクニックである，ストップワードの削除，自立語の区別，パラメータの調整は行わなかった．

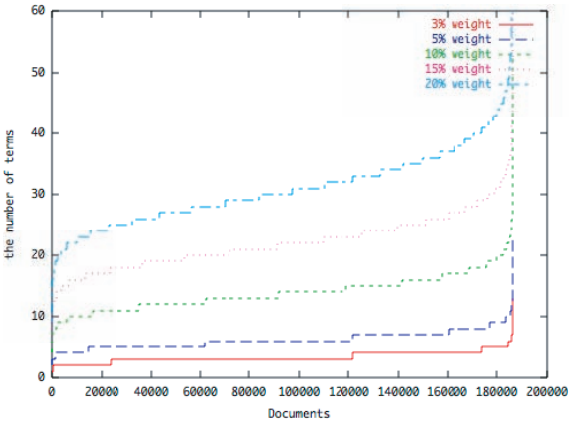

図 3　文書におけるキーワードの重みの総和の一定割合を占めるのに最低限必要なキーワード数

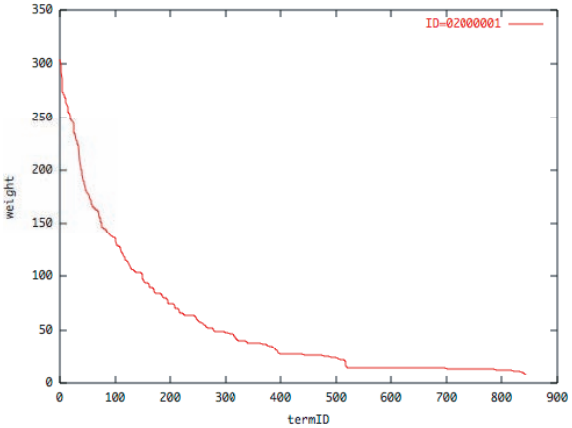

図 4　キーワードの重みの分布 (ID=02000001)

文書におけるキーワードの重みの総和の一定割合を占めるのに最低限必要なキーワード数を示したものが図 3 である．そして，文書におけるキーワードの重みの分布について個別の文書について見たものが図 4 である．データとして 2002 年度前期の日本国特許公報全文に含まれる，ID が 02000001 のファイルを用いた．

図 4 のキーワードの重みの分布から，特徴あるキーワードからそうでないものまで幅広いキーワードが出現していることがわかる．図 3 からは文書におけるキーワードの重みの総和の 5%を占めるには平均 6.14 個のキーワードが必要なことがわかった．これはつまり，1000 個に分散符号化したうちの 50 個の分散情報から復元を可能にした場合，平均 6.14 個の重みの大きいキーワードを含むクエリを生成して投げたとき、文書を復号するのに十分な分散情報を取得できることを意味する．

検索クエリを投げたときにアクセスする可能性の高いノードほど多くの分散情報を保持していると，ファイルの収集効率は向上する．共有目的のファイルに対して重みの大きいキーワードほどクエリに含まれやすいという仮定が正しければこの配分方法は効率が良いと言える．

なお，本研究では過去に投げたクエリにより得られた分散情報もローカルに適当な量を保持する構造になっている．そのため，実際にキーワードを投げて得られる分散情報に加えて，ユーザの興味分野の分散情報も幾分か保持している状況が自然であると考えられるので，多少キーワードが少なくてもある程度の復元が可能であると思われる．また，(k, n) 閾値法のパラメータを変更することで，蓄積コストも大きくなってしまうが，収集に必要なキーワード数を減らすことができる．

5. おわりに

本稿では，インデックスの作成とデータの配置，検索，復元の 3 ステップで構成される，P2P 情報検索における索引とファイルの分散配置手法，Concordia について述べた．具体的には，(k, n) 閾値法を用いてファイルを分散符号化し，ファイルにおけるキーワードの重みに応じてそれに対応付けする分散情報の数を決め，DHT 上に配置を行う手法である．本手法は，ノードの参加離脱が頻繁で，ネットワークの規模が大きく，同一ファイルを複数のノードで所持していたりノードのもつ情報が偏っていないような状況において有効であり，個々のファイルのクエリとの適合度を考慮した検索や効率的な冗長化が可能である．

現在は，少ない有用なクエリに対してさらに効率よく収集可能にすべく，ローカルに持つ情報のみを用いた軽量なクエリ拡張などを検討中である．また，検索応答時間を向上させるために，ピアのネットワーク環境を踏まえたファイル配置ができるように改善を検討している．

文　献

[1] Google. http://www.google.com/

[2] MeCab. http://mecab.sourceforge.jp/

[3] NTCIR. http://research.nii.ac.jp/ntcir/index-en.html

[4] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. MINERVA: Collaborative P2P Search. In Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05), 2005.

[5] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Improving Collection Selection with Overlap Awareness in P2P Search Engines. In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05), pp.67-74, 2005.

[6] B. H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. Commn. ACM, Vol.13, No.7, pp.422-426, 1970.

[7] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In Proceedings of the 12th International Symposium on High Performance Distributed Computing (HPDC '03), 2003.

[8] H. Fang, T. Tao, and C. Zhai. A formal study of information retrieval heuristics. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '04), pp.49-56, 2004.

[9] C. Gkantsidis, and P. Rodriguez. Network coding for large scale content distribution. In Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05), 2005.

[10] V. Gopalakrishnan, B. Silaghi, B. Bhattacharjee, and P. Keleher. Adaptive Replication in Peer-to-Peer Systems. In Proceedings of the 24th International Conference on Distributed Computing Systems, 2004.

[11] P. Lyman, H. R. Varian, K. Swearingen, P. Charles, N. Good, L. L. Jordan, and J. Pal, "How much information 2003?", http://www.sims.berkeley.edu/research/projects/how-much-info-2003/, 2003.

[12] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In Proceedings of IPTPS'02, 2002.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01), pp.161-172, 2001.

[14] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock Beaulieu, and M. Gatford. Okapi at TREC-3. In Proceedings of TREC-3, pp.109-126, 1994.

[15] A. Shamir. How to share a secret. Commn. ACM, Vol.22, Num.11, 1979.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01), pp.149-160, 2001.

[17] C. Tang, and S. Dwarkadas. Hybrid Global-Local Indexing for Efficient Peer-To-Peer Information Retrieval. In Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI), 2004.

[18] H. Yamamoto. On secret sharing systems using (k,L,n) threshold scheme. IECE Trans. Vol.J68-A No.9 pp.945-952, 1985.

[19] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical report, U. C. Berkeley Technical Report USB//CSD-01-1141, 2001.