

修士論文

文書情報を利用した  
P2P 情報検索の効率化に関する研究

( A Study to Improve Peer-to-Peer  
Information Retrieval Schemes Using Document Features )



倉沢 央

東京大学大学院 情報理工学系研究科 電子情報学専攻

指導教員 安達 淳

2008 年 2 月 4 日提出

# 概要

膨大な情報を効率よく扱うため、検索システムが広く一般的に使われている。検索システムの多くは集中型のアーキテクチャで構築されている。この集中型のアーキテクチャは容易に構築でき、さらに情報を集中管理できる。そのため、小規模なシステムやデータマイニングを伴うシステムに集中型のアーキテクチャは適している。しかしながら、検索システムの規模が大きくなるにつれ、スケーラビリティやシステム管理コストの面で集中型の検索システムを運営することは困難になっていく。大規模検索システムの構築や運営には、負荷分散についての専門知識や、膨大なコンピューティング資源、豊富な資本力が必要となるからである。

これらの問題を解決できる手法として、分散型のアーキテクチャが注目を浴びている。分散型のアーキテクチャは、大規模システムへの対応だけでなく、余剰コンピューティング資源の有効活用やシステムの柔軟性の確保の点でも期待されている。これまでの研究で、分散型検索システムの検索精度を集中型とほぼ同等となる手法が提案されている。しかし、分散型検索システムにおける索引構築コストや検索実行時のコストが未だボトルネックとなっている。筆者は P2P 情報検索を実用的なものにすべく、これら問題を解決する索引構造とデータ配置法に関する 2 つの手法を提案する。

1 つめの手法は Huffman-DHT である。Huffman-DHT は筆者の提案する P2P 情報検索のための新たな索引構造である。Huffman-DHT は索引登録時のホットスポットを解消し、さらに索引構築に伴う探索コストを低減することを目的とする。これを実現するため、登録頻度の高い一部の単語の索引が索引構築コストの大部分を占めているという Zipf の法則に基づいた特徴を利用し、Huffman-DHT では出現確率の高い単語に対して ID 空間で広い領域を割り当てる。ID 空間の分割には、符号理論の Huffman 符号を採用した。Huffman-DHT を用いることで、索引構築の際の被アクセス数をノード間で分散できる。さらに、通常の索引では単語の索引を保持するノードを探すのにその単語の出現確率に関わらず  $O(\log n)$  のホップ数が必要となるところを、登録頻度の高い単語の索引ほど少ないホップ数で探せられる。

もう1つの手法は Concordia である．Concordia は筆者の提案する P2P 情報検索のための新たなデータ配置手法である．Concordia は，P2P 情報検索において P2P ネットワークからユーザの検索問い合わせに適合する文書を効率良く高速に収集することを目的とする．これを実現するため，Concordia では文書データの配置場所を検索時に索引参照のために接続する場所と関連づける．つまり，文書と関連度の高い索引を管理するノードに文書データを配置する．その関連度の計算には文書中の各単語の重みを用いる．Concordia を用いることで，問い合わせに適合する文書ほど収集が容易となり，P2P 情報検索の実行時間を削減できる．

本論文ではこれら2つの手法について述べる．筆者は分散型検索における課題を克服し，実用的なものにすることを研究目標と据えている．

# 目次

<b>第 1 章</b>	<b>序論</b>	<b>1</b>
1.1	研究の背景と目的 . . . . .	3
1.2	本論文の構成 . . . . .	5
<b>第 2 章</b>	<b>関連研究</b>	<b>6</b>
2.1	Peer-to-Peer アーキテクチャ . . . . .	7
2.1.1	Hybrid P2P . . . . .	7
2.1.2	Pure P2P . . . . .	9
2.2	P2P 環境における情報検索手法 . . . . .	14
2.2.1	検索アルゴリズムの評価法 . . . . .	14
2.2.2	Unstructured P2P における検索手法 . . . . .	14
2.2.3	Structured P2P における検索手法 . . . . .	16
2.2.4	P2P 環境における検索手法の比較 . . . . .	20
2.3	P2P 環境における負荷分散手法 . . . . .	22
2.4	P2P 環境におけるデータ配置手法 . . . . .	24
2.5	既存研究の問題点と本研究での取り組み . . . . .	25
<b>第 3 章</b>	<b>Huffman-DHT: 単語の出現確率に基づいた索引構造</b>	<b>26</b>
3.1	Huffman-DHT の概要 . . . . .	27
3.2	Huffman-DHT のアーキテクチャ . . . . .	29
3.2.1	Huffman-DHT の構造 . . . . .	29
3.2.2	ノードの管理する ID . . . . .	30
3.2.3	ノードの探索 . . . . .	30
3.2.4	文書の索引への登録 . . . . .	31
3.3	Huffman-DHT の評価 . . . . .	33
3.3.1	単語の出現確率分布の推定精度 . . . . .	33
3.3.2	索引付けの際のホップ数 . . . . .	34
3.3.3	索引登録時の被アクセス数の分散度合い . . . . .	42

---

3.3.4	索引の同期が必要なノード数 . . . . .	44
3.4	Huffman-DHT のまとめと今後の課題 . . . . .	46
<b>第 4 章</b>	<b>Concordia: 単語の重みに基づいたデータ配置手法</b>	<b>47</b>
4.1	Concordia の概要 . . . . .	48
4.2	Concordia のアーキテクチャ . . . . .	49
4.2.1	DHT を用いた索引構造 . . . . .	49
4.2.2	単語の重み付けと索引への登録 . . . . .	49
4.2.3	単語の重みに基づいたデータ配置 . . . . .	51
4.2.4	検索とデータの収集 . . . . .	52
4.2.5	Erasure 符号を用いた文書の分割法 . . . . .	53
4.3	Concordia の評価 . . . . .	55
4.3.1	総文書数の設定が検索性能へ及ぼす影響 . . . . .	55
4.3.2	索引ノードのみから得られる文書の検索精度 . . . . .	56
4.3.3	適合文書の収集時に接続するノード数 . . . . .	57
4.3.4	各文書における単語の重みを用いる有効性の検証 . . . . .	59
4.3.5	検索応答時間 (PC クラスタを用いた実証実験) . . . . .	61
4.3.6	Erasure 符号の有効性 . . . . .	62
4.4	Concordia のまとめと今後の課題 . . . . .	69
<b>第 5 章</b>	<b>結論</b>	<b>70</b>
	謝辞	73
	参考文献	75
	発表文献	79

# 図目次

1.1	Huffman-DHT と Concordia	2
1.2	情報爆発時代 出処：特定領域研究『情報爆発時代に向けた新しい IT 基盤技術の研究』	4
2.1	クライアント・サーバ型ネットワークと P2P ネットワークの比較	7
2.2	幅優先探索	10
2.3	深さ優先探索	11
2.4	Chord	12
2.5	Kademlia	13
2.6	クラスタ形成タイプ	16
2.7	Summary 共有タイプ	17
2.8	文書 ID タイプ	18
2.9	term-to-peer 索引タイプ	19
2.10	term-to-document 索引タイプ	20
2.11	P2P 環境における索引付け手法の比較	21
3.1	Huffman-DHT の符号木の構築方法	30
3.2	Huffman-DHT における単語 ID の割り当て方法	31
3.3	Huffman-DHT におけるノードの管理する ID	32
3.4	文書コレクションにおける Document Frequency の分布	35
3.5	頻度の高い単語の DF 値が全体で占める割合	36
3.6	出現確率の推定値と実際の値との類似度	37
3.7	索引付けの際のホップ数 (理論値)	41
3.8	索引付けの際のホップ数 (シミュレーション)	43
3.9	索引登録時の被アクセス数の分散度合い	44
3.10	索引の同期が必要なノード数の最大値	45
4.1	索引付けとデータ配置法 (Concordia-1)	52

---

4.2	索引付けとデータ配置法 (Concordia-2) . . . . .	53
4.3	検索と文書データの収集 (Concordia-2) . . . . .	54
4.4	TREC-2 ad hoc & TREC-3 routing topics の一例 . . . . .	56
4.5	総文書数の推定値を変化させたときの Interpolated Recall- Precision Average . . . . .	57
4.6	索引ノードのみから得られる文書の検索精度 (Concordia-1) . . . . .	58
4.7	索引ノードのみから得られる文書の検索精度 (Concordia-2) . . . . .	59
4.8	適合する上位 n 文書の収集時に接続するノード数 (Concordia-1) . . .	60
4.9	適合する上位 n 文書の収集時に接続するノード数 (Concordia-2) . . .	61
4.10	各単語についてのストレージコストの比較 . . . . .	64
4.11	適合する上位 n 文書の収集時に接続するノード数の比較 . . . . .	65
4.12	適合する上位 n 文書の収集時間 (Concordia-1) . . . . .	66
4.13	Erasure 符号の分割と復号に必要な計算時間 ( (20,20,100) 分割 ) . . .	66
4.14	Erasure 符号の分割と復号に必要な計算時間 ( 1MB のファイルを (N/5,N/5,N) 分割 ) . . . . .	67
4.15	単語の重みの分布 (ZF32-345-1050) . . . . .	68
4.16	文書のデータの損失確率 (ZF32-345-1050) . . . . .	68

# 表目次

2.1	クライアント・サーバ型ネットワークと P2P ネットワークの比較 .	8
4.1	配置法ごとの文書の複製数の比較 . . . . .	62
4.2	実験で用いた InTrigger マシンの詳細 . . . . .	63
4.3	冗長化手法の比較 . . . . .	65

# 第 1 章

## 序論

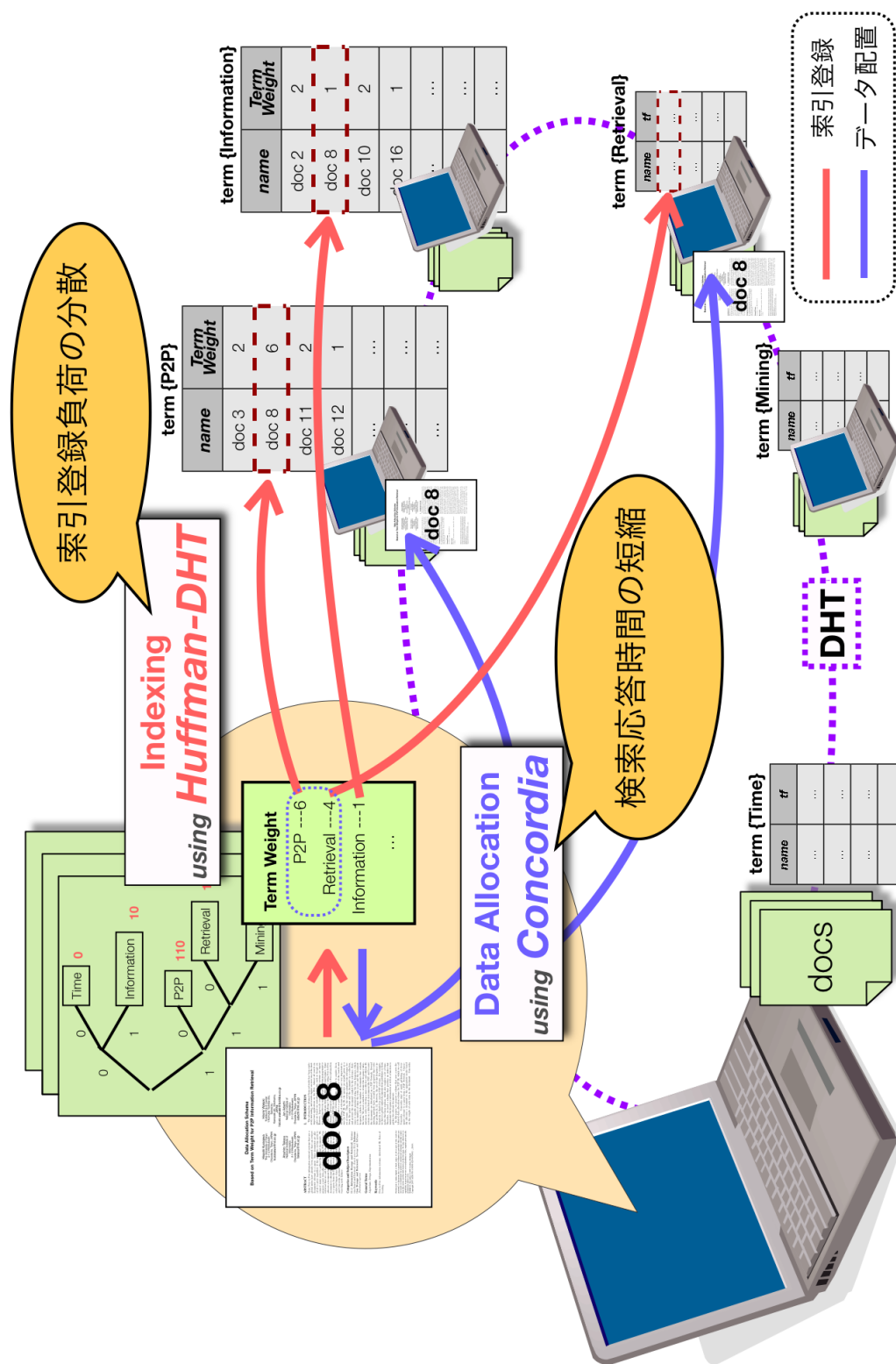


図 1.1: Huffman-DHT と Concordia

## 1.1 研究の背景と目的

インターネットの登場により人類の生産する情報は爆発的に増加し続けている [LVS<sup>+</sup>03]。図 1.2 からわかるように、年々その増大の勢いを増している。このような状況を背景に、検索システムは日常生活における重要なツールの 1 つとなった。検索システムを利用することで、膨大な情報の中から欲しい情報をそのキーワードから効率良く探せる。Web 上のテキスト、ニュース記事、地図、乗り換え情報、画像、人、薬品の効能、プログラムのソースなど、世界中の情報が整理され、検索システムからアクセスできるようになりつつある。今後さらに検索システムを経由してアクセスできる情報は増えると予想される。それと同時に、検索システムは日々の生活にますます欠かせないものになっていくと思われる。

従来、検索システムの多くは集中型のアーキテクチャで構築されている。この集中型のアーキテクチャは容易に構築でき、さらに情報を集中管理できる。そのため、小規模なシステムやデータマイニングを伴うシステムに最適である。しかしながら、システムの規模を大きくしようとする、集中型のシステムは以下に挙げる点がボトルネックとなる。

- スケーラビリティ
- 検索対象の網羅性
- 管理・維持コスト

既存の大規模検索サービスではこれらの問題点に対して、高度な専門知識や膨大なコンピューティング資源、豊富な資本力で解決を図っている。一方、収益性の少ないサービスや一般ユーザが主体となっているサービスでは、これらの解決法は難しい。

これに対して、大規模なシステムに向いている分散型アーキテクチャで検索システムを構築することを目指した研究が近年盛んになりつつある。Peer-to-Peer(P2P)ネットワークを用いることで、参加するユーザ(ノード)が多ければ多いほど大規模な検索システムの運営が可能となる。つまり、収益性がなくても利用者が多ければ、膨大な情報を P2P 情報検索システムで整理できる。P2P 情報検索は大規模システムへの対応だけでなく、更新頻度の高い情報の共有や余剰コンピューティング資源の有効活用、システムの柔軟性の確保の点でも期待されている。これまでの研究で、分散型検索の検索精度は集中型とほぼ同等となる手法が提案されて

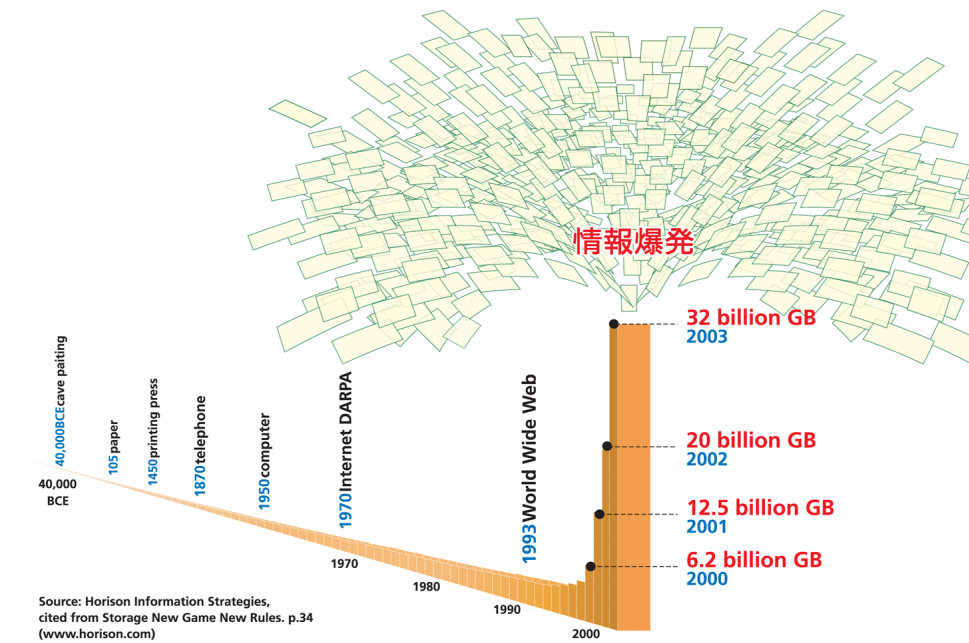


図 1.2: 情報爆発時代 出処：特定領域研究『情報爆発時代に向けた新しい IT 基盤技術の研究』

いる．しかし，分散型検索システムにおける索引構築のオーバーヘッドや検索実行時のコストが未だボトルネックとなっている．

そこで筆者は，P2P 情報検索を実用的なものにすべく，これら問題を解決する索引構造とデータ配置法を提案する．本論文で提案する索引構造である Huffman-DHT は，ノードの索引に関わる負荷を均一にすることを目的とした．Huffman-DHT では，登録頻度の高い一部の単語の索引が索引構築処理の大部分を占めているという Zipf の法則に基づいた特徴を利用し，出現確率の高い単語に対して ID 空間で広い領域を割り当てる．Huffman-DHT を用いることで，索引登録時のホットスポットを解消し，さらに索引構築に伴う探索コストを低減した．また，提案するデータ配置法である Concordia は，P2P 情報検索において P2P ネットワークからユーザの検索問い合わせに適合する文書を効率良く高速に収集することを目的とした．Concordia では，Concordia では文書データの配置場所を検索時に索引参照のために接続する場所と関連づける．Concordia を用いることで，問い合わせに適合する文書ほど収集が容易となり，P2P 情報検索の実行時間を削減した．本論文で提案する 2 つの手法はいずれも P2P 情報検索のための基盤技術である．Huffman-DHT と Concordia の概要を表したものが図 1.1 である．

## 1.2 本論文の構成

本稿では、第2章にてP2PアーキテクチャとP2P環境における検索手法について説明する。第3章では、P2P情報検索における索引構造である Huffman-DHT を提案する。また、第4章では、P2P情報検索におけるデータ配置法である Concordia を提案する。最後に、第5章にて今後の課題とともにその発展の方向性を示唆する。

## 第 2 章

### 関連研究

## 2.1 Peer-to-Peer アーキテクチャ

Peer-to-Peer(P2P) ネットワークとは、不特定多数のノード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TCP/IP より上位のアプリケーション層で相互に接続し、仮想的なネットワークを形成している。クライアント・サーバ型ネットワークと P2P ネットワークを比較したものが表 2.1 である。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。クライアント・サーバ型ネットワークと P2P の 2 つのアーキテクチャの構造を示したものが図 2.1 である。それぞれのアーキテクチャの構造と検索機能について述べる。

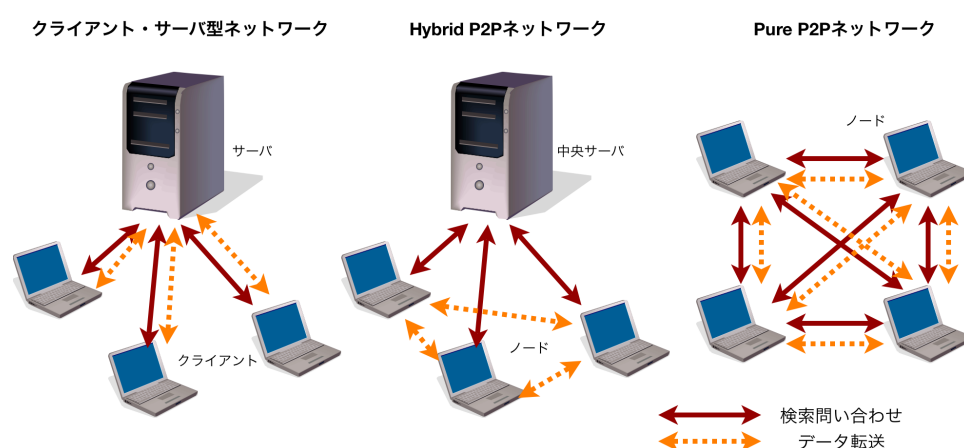


図 2.1: クライアント・サーバ型ネットワークと P2P ネットワークの比較

### 2.1.1 Hybrid P2P

Hybrid P2P アーキテクチャは、クライアント・サーバモデルを残した P2P の形態であり、Napster[nap] など初期の P2P ファイル共有ソフトウェアで用いられていた。接続されているノードのアドレスやすべてのデータの配置場所の情報は中央サーバで管理する。一方、ファイル自体は各ノードで管理するモデルとなっている。データを検索・収集するたびに中央サーバを経由させられるため、セキュリティ管理、認証を行いやすいという利点を持つ。ファイルの検索には中央サーバの索引を利用するため、検索応答は一般に Pure 型より速い。しかし、ノード数、ファイル数の増加に伴い検索処理は遅くなる。また、常に新しい情報を検索できる保証はない耐障害性においては、特定の機能を 1 ヶ所のサーバに頼っているた

表 2.1: クライアント・サーバ型ネットワークと P2P ネットワークの比較

	クライアント・サーバ型ネットワーク	P2P ネットワーク
モデル	サーバとクライアントの 2 種類のコンピュータ	サーバとクライアントの 2 つの機能を 併せ持つコンピュータ
スケーラビリティ	△ (ロードバランシング, DNS の利用, Web キャッシュ などのアプローチでシステム増強)	○ (ノードの追加)
データの制御・管理	○ (データはサーバにて集中管理)	△ (データはネットワーク上に分散)
耐障害性	× (サーバへリスクの集中)	○ (他の Peer で代替可能)
アドホック性	× (端末構成の変更が困難)	○ (端末の構成を自由に変更可能)

め、ここが単一障害点になりやすいという欠点を持つ。検索機能だけに注目すると、集中型検索システムの構造と似ている。

### 2.1.2 Pure P2P

Pure P2P アーキテクチャは、Gnutella[gnu] や Freenet[CSWH01] に代表されるタイプであり、Hybrid P2P 型と異なりサーバを必要とせず、純粹にノード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ノードのうち一つとまず接続し、そのノードを介して他のノード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。Pure P2P はノードの参加や離脱にも自律的に適応できるため、ノード数やファイル数に対して高いスケーラビリティをもつと同時にアドホック性も高い。また、Hybrid 型と異なり、中央サーバを持たない。中央サーバなしでデータを検索・収集できるため、管理や認証は組み込みにくいという欠点があるその一方で、匿名性を向上させる機能を持たせやすい。さらに、中央サーバという単一障害点を持たないため障害に強い。検索機能に注目すると、すべてのファイルやノードを管理する機構がないため、手法が複雑である。手法によってはノードの増加に伴いトラフィックの増加を招くこともあり、現在様々な研究がなされている。情報の伝搬方法や管理方法、検索手法から、Unstructured 型と Structured 型の 2 つに分けることが出来る。

#### Unstructured P2P

Unstructured P2P は Pure P2P アーキテクチャにおけるネットワーク構造の 1 つである。Unstructured P2P は、隣接ノードについての制約がない。そのため、ネットワークを形成する過程で生じるトラフィックは小さい。オブジェクトの発見は隣接ノードへ検索クエリを転送する手法を採るため、検索時に生じるトラフィックは大きい。Unstructured P2P は検索効率は悪いが、瞬時にネットワークを構築でき、さらに検索問い合わせの内容を柔軟にできることが特徴である。Unstructured P2P における代表的な 2 つの検索手法について説明する。

**幅優先探索** Gnutella[gnu] では、隣接ノードへ検索クエリを送ることを繰り返し、幅優先探索によってファイルを探す。検索条件に一致するファイルが見つかったら、ファイルを保持するノードのアドレス情報を、検索クエリの経路を逆にたどることで転送する。受信したアドレス情報をもとにファイルの交換を直接行う。近隣

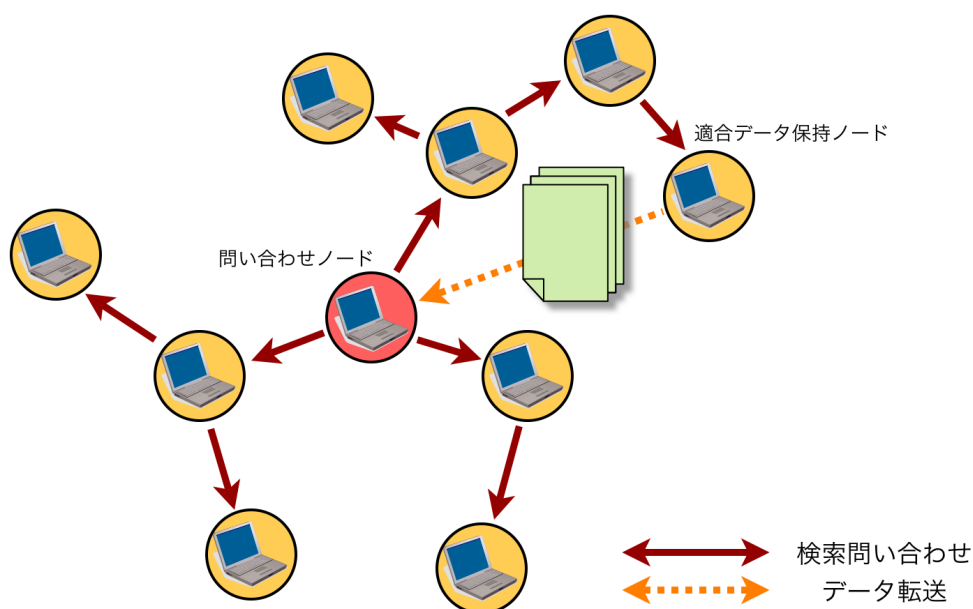


図 2.2: 幅優先探索

ノードへ送信するとき、洪水のようにパケットをネットワークに流すので、この手法は『フラッディング』と呼ばれている。トラフィックの軽減のため、同一の検索問い合わせを伝搬にしないように、各検索クエリにはUUID(Universally Unique Identifier) もしくは GUID(Globally Unique Identifier) と呼ばれる MAC アドレスと時刻と乱数が組み合わされた一意な識別子が設定されている。また、TTL(Time To Live) と呼ばれる転送カウンタの設定により無限に問い合わせが転送されないようにしている。Gnutella はファイル数とノード数についてはスケーラビリティは高いが、トラフィックの面ではスケーラビリティは高いとは言えない。幅優先探索を示したものが図 2.2 である。

**深さ優先探索** Freenet[CSWH01] では、隣接ノードのうち有力なノードに転送することを繰り返し、深さ優先探索によってファイルを探す。ファイル名、ノードのアドレスとともにハッシュ値が利用され、有力なノードはハッシュ値の近さによって判断される。検索条件に一致したファイルが見つかったら、ファイル保持ノードのアドレス情報ではなくファイル本体が経路を逆にたどり転送される。経路の途中のノードではそのファイルをキャッシュする。匿名で受信も送信も行えること、需要の多い情報は多くのノードのキャッシュに残り意図的に消すことができないことが特徴として挙げられる。つまり、キャッシュ機能によってニーズに基づいた情報の複製がされているとも言える。Freenet はトラフィックの軽減、ニーズの高

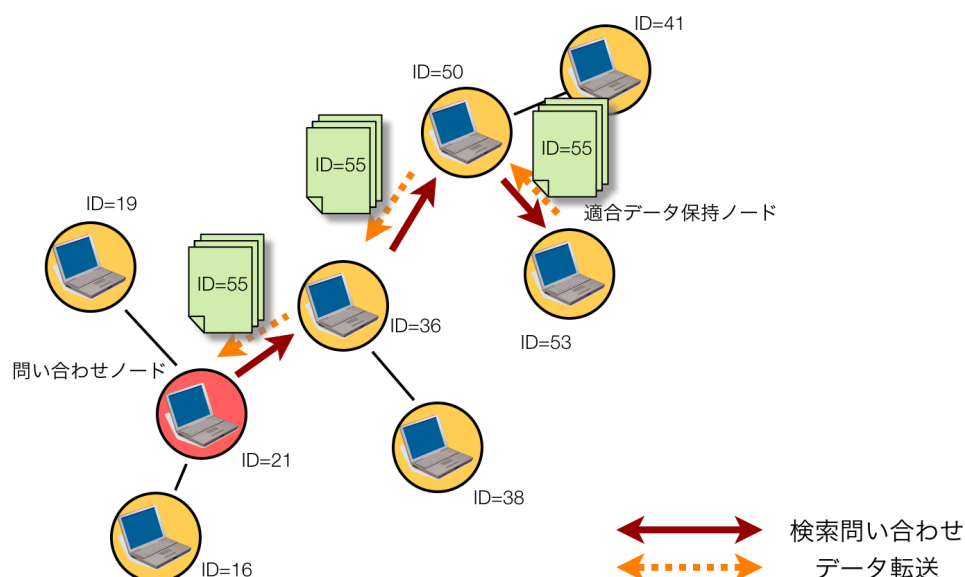


図 2.3: 深さ優先探索

いファイルのクラスタ化ができ、効率良く情報を伝搬できる。深さ優先探索を示したものが図 2.3 である。

### Structured P2P

Structured P2P は Pure P2P アーキテクチャにおけるネットワーク構造の 1 つで、DHT(Distributed Hash Table) によって構築されたネットワークを言う。DHT(Distributed Hash Table) では、ノードのアドレスとファイルは共通のハッシュ関数で生成される ID で管理される。ノードは、自身の ID と近い ID のファイルを保持し、また、隣接ノードも ID で決められる。この ID に基づいた制約があるため、ネットワークの構築には多少時間を要する。オブジェクトは ID をもとに検索クエリを最適なノードへ転送することで発見するため、検索時に生じるトラフィックは小さい。Structured P2P は DHT に基づいたネットワークを構築する手間はあるが、効率良く検索できることが特徴である。DHT の代表的な例として、円構造の Chord[SMK<sup>+</sup>01]、木構造の Tapestry[ZKJ01] や Kademlia[MM02]、多次元空間の CAN[RFH<sup>+</sup>01]、SkipGraph[AS07] などが挙げられる。これらのうち、Chord と Kademlia について説明する。

**Chord** Chord では、 $N$  個のオブジェクトからなる円構造のハッシュ空間を形成する。 $N$  の最大値は  $2^{scale}$  である。各ノードは、前後に位置するノードの ID、管

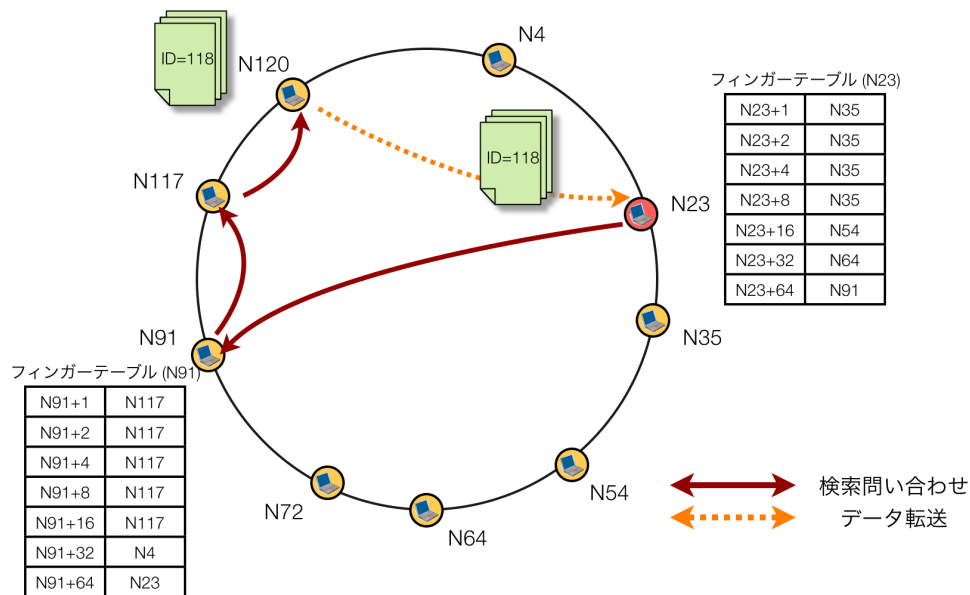


図 2.4: Chord

理するデータの ID リスト, そしてフィンガーテーブルを保持する.  $j$  番目に位置する ID が  $i$  のノード  $i$  を仮定すると, ノード  $i$  は  $j-1$  番目のノードの ID から  $i$  の間に該当する ID のファイルを担当する. フィンガーテーブルには, ID が  $i+2^k$  ( $k = 0, 1, 2, \dots, scale-1$ ) のオブジェクトを管理するノードのリンク情報が書かれている. オブジェクトの検索の際には, フィンガーテーブル内で最もオブジェクトの ID に近いノードまでアクセスすることを繰り返す. Gnutella 型ではノード数  $N$  の増加に伴い線形にパケットは増加するが, Chord ではパケット増加は  $O(\log N)$ , ホップ数は  $O(\log N)$  となり, スケーラビリティが高い. オブジェクトの検索速度は  $O(\log N)$  となる. Chord の検索手法を示したものが図 2.4 である.

**Kademlia** 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 の計算値で大きい桁から見て 1 が出現するビットの位置で決まる. つまり 2 分木構造となっている. 各ノードは, 自身との距離ごとに最大  $k$  個の他ノードのアドレスを保持する  $k$ -buckets と呼ばれる経路表を持つ. 160bit でノード ID を表現している場合, 経路表には自身との距離が  $[2^i, 2^{i+1}]$  ( $0 < i \leq 160$ ) であるアドレスがまとめられる. XOR を使うことによりノード同士の距離が容易にわかるので, いかなるメッセージのやり取りの際でも経路表を更新できる. 何かしらのメッセージを他ノードから受け取ると, そのノードの情報を経路表の最後尾に追加する. つまり, 経路表には追加した順番でノード情報が格納されている. もし経路表に  $k$  個のノード情報で満たされている場合は, 経路表の先頭に格納

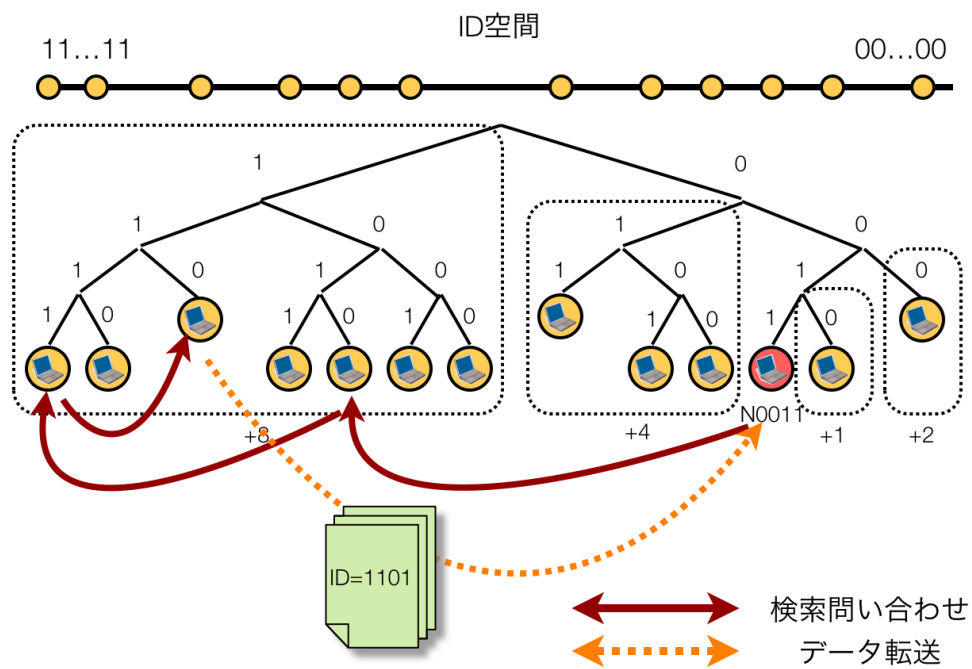


図 2.5: Kademlia

されているノードの生存確認を行う。そのノードがまだ通信可能な場合、先頭から最後尾に入れ替え、新しく受け取ったノード情報は破棄する。これにより長く生存しているノードの情報を保持し続けることができ、ノードの頻繁な参加、離脱が起きてもネットワークを維持できる。検索の際には、オブジェクトのIDと距離の近い  $k$  個のノードアドレスを得るまで、長さ  $k$  のノードアドレスリストを更新していく。検索速度は  $O(\log N)$  となる。Kademlia の検索手法を示したものが図 2.5 である。

## 2.2 P2P 環境における情報検索手法

P2P 環境における情報検索手法について説明する。情報検索システムは、ユーザからの問い合わせに適合する文書を探しだし、これら文書を取得したいという要求を満たすためのシステムである。P2P 情報検索では、P2P 環境において問い合わせに適合する文書をいかに効率良く探し出し、そして収集するかが重要な課題である。情報検索における問い合わせは複数の単語から成る。情報検索システムでは問い合わせと蓄積されている文書間の適合度を計算し、適合度の大きい文書をユーザに提示する。つまり、従来の検索要求に一致するオブジェクトを単純に探し出す手法だけではなく、適合度を考慮した提示も求められる。

本節ではまず、検索アルゴリズムの評価法について述べる。その後、Unstructured P2P と Structured P2P それぞれにおける検索手法を紹介する。最後に、紹介した手法を比較し、本論文が提案する手法の既存研究における位置づけを説明する。

### 2.2.1 検索アルゴリズムの評価法

情報検索の分野では、システムの評価は一般に以下の 2 つの評価尺度が最も一般的に用いられている。

- 再現率 (recall) : 完全性、すなわち検索漏れの少なさを評価するための尺度であり、検索対象となる文書集合の中の検索質問に適合する文書のうち、実際に検索された文書の割合を示す。

$$\text{再現率} = \frac{\text{検索された文書中の適合文書の数}}{\text{全文書中の適合文書の数}}$$

- 適合率 (precision) : 正確性、すなわち検索ノイズの少なさを評価するための尺度であり、検索された文書集合の中で、検索質問に適合する文書の割合を示す。

$$\text{適合率} = \frac{\text{検索された文書中の適合文書の数}}{\text{検索された文書の数}}$$

### 2.2.2 Unstructured P2P における検索手法

Unstructured P2P における検索手法は、適合度の算出だけでなく、検索問い合わせの処理で発生するトラフィックを抑制する工夫がされている。従来の幅優先探索

もしくは深さ優先探索では、検索問い合わせのたびにネットワークへ大きな負荷を引き起こす。そこで関連研究では、検索問い合わせと関連度の高いノードに対して検索問い合わせを転送できるように工夫している。本節では関連研究を、クラスタ形成タイプ、term-to-peer 索引タイプの2つに分類して紹介する。

**クラスタ形成タイプ** Unstructured P2P では隣接ノードについての制約がないことを利用し、クラスタ形成タイプでは、保持する文書が類似したノードと積極的に接続することでクラスタを形成し、検索効率を向上させている。

GES では、ノードの持つ文書からベクトルを作成する [ZYH05]。各ノードは、類似したベクトルを持つノードとのリンクとそれ以外のノードとのリンクを区別して保持する。検索時には、検索問い合わせを隣接するノードのうちの1つに転送していく。その過程で問い合わせと適合する文書を保持するノードにたどり着いたら、そのノードを起点に類似したベクトルを持つノードへ検索問い合わせをフラッディングする。この2つの転送手法を組み合わせることで検索効率を向上させている。しかし、GES ではノードの持つ文書に複数トピックが含まれていても、ノードを示すベクトルではそれらトピックが混在して扱われてしまう欠点がある。そこで CSS では、各ノードは文書をトピックの異なるクラスへ分類して保持する [HLW07]。ノード間の物理的なリンクを保持する際に、ノードの持つクラスの類似度を比較して最短の距離と最長の距離の2つのリンクを求める。そうすることで、ノードの持つ複数のトピックを別に扱えるようにしている。

クラスタ形成タイプは、ノードが持つトピックに偏りがある状況で効果を発揮する。しかし、検索性能の再現率を高めることが難しいという欠点を持つ。クラスタ形成タイプの検索手法を示したものが図 2.6 である。

**Summary 共有タイプ** Summary 共有タイプではノードの持つ文書情報を軽量にすることで伝搬しやすくし、検索精度を向上させている。

PlanetP では、単語とノードアドレスを対応づける Bloom Filter [Blo70] を用いた文書情報の Summary をネットワークに拡散・同期することで term-to-peer の索引を構築した [CAPMN03]。他ノードへの Summary の拡散には Gossiping と呼ばれるアルゴリズムを使用している。PlanetP における検索は、まず探したい単語を含む文書を持つノードを、全ノードで同期している Summary から探す。そして絞り込んだノードに対してクエリを投げ、それぞれのノードで文書のランクを計算し、適切な文書とされた URL を取得する。文書のランク付けには Vector space

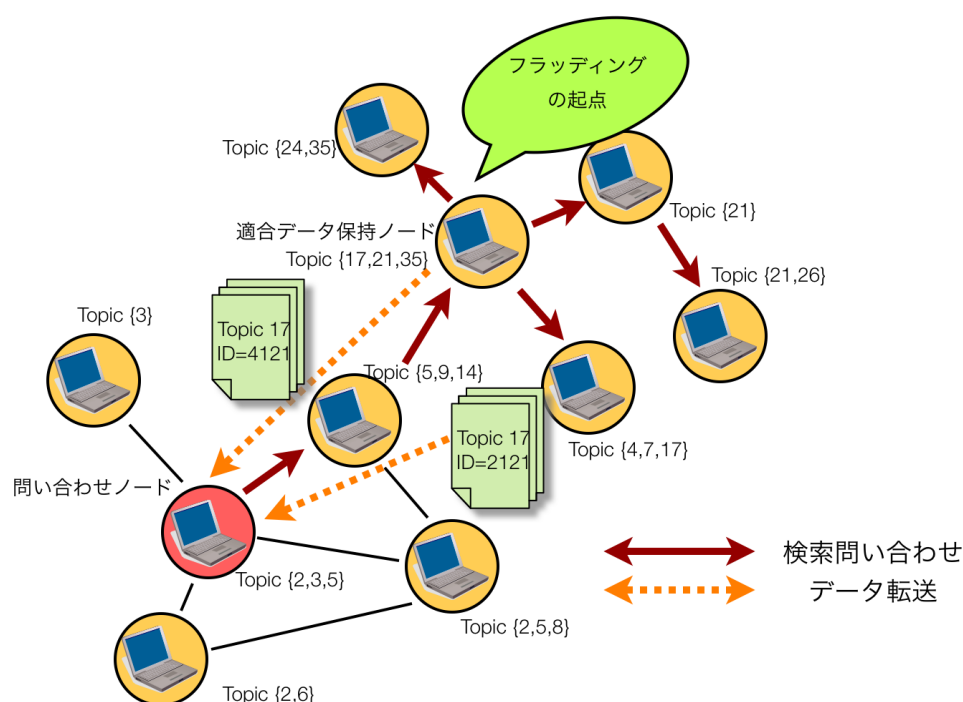


図 2.6: クラスタ形成タイプ

ranking model の  $TF \cdot IDF$  を元にしたアルゴリズムが用いられている． $IDF$  の代わりに  $IPF$  (Inverse Peer Frequency) を新たな尺度として定義し，単語の大域的な重み付けを求めている．以上のようにして PlanetP ではクエリを転送することなく適合文書を持つノードを容易に絞り込める．しかし，PlanetP はノードの増大に伴い Summary の伝搬トラフィックが増える欠点がある．そこで Rumorama では，ノードに ID を割り振り小さいグループに分割することでスケーラブルなシステムに改良した [MEH05]．Rumorama は隣接ノードとの接続を階層化することで検索問い合わせの伝搬や Summary の拡散を効率化させている．

Summary 共有タイプはすべてのノードの持つ文書の特徴を容易に得られるという利点がある．しかし，索引のみから個々の文書を特定することができないという欠点を持つ．Summary 共有タイプの検索手法を示したものが図 2.7 である．

### 2.2.3 Structured P2P における検索手法

Structured P2P における検索手法は，いずれも DHT 上で文書に割り当てられる ID を工夫している．本節では既存研究を，文書 ID タイプ，term-to-peer 索引タイプ，そして，term-to-document 索引タイプの 3 つに分類して紹介する．

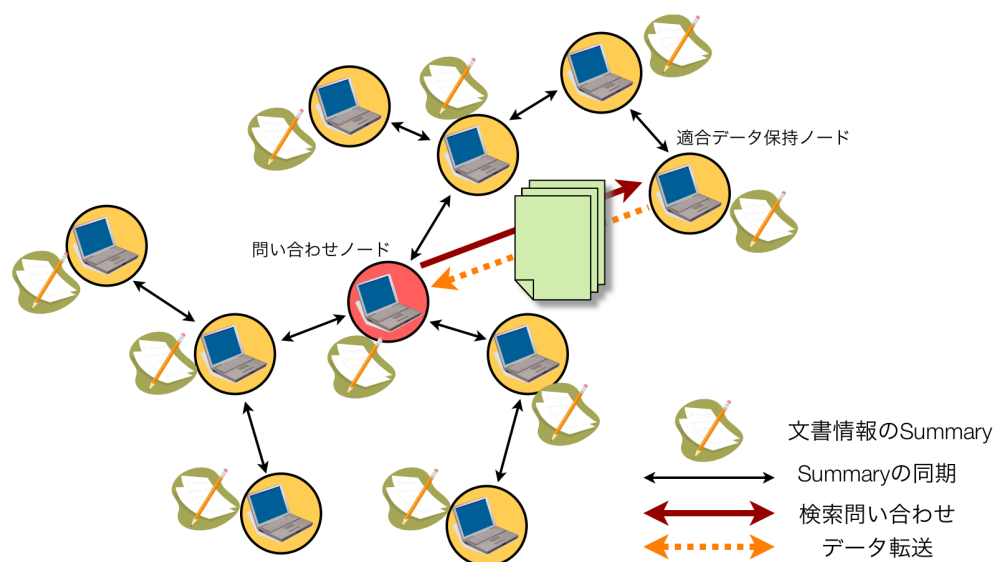


図 2.7: Summary 共有タイプ

**文書 ID タイプ** 文書 ID タイプでは、文書における単語ベクトルを DHT のハッシュ空間に投影することで、文書に対して ID を割り振っている。つまり、文書に割り当てられる ID は 1 つである。

pSearch では、文書の単語ベクトルの次元を Latent Semantic Indexing (LSI) を用いて落とし、CAN のハッシュ空間上に配置した [TXD03]。そうすることで、同じようなトピックの文書に対して近い ID を割り当てる。検索の際には、検索問い合わせの単語ベクトルを LSI によって定め、DHT 空間で近い ID の文書を探す。文書の ID が文書の内容に基づいたベクトルなので、Vector Space Model (VSM) で文書の問い合わせとの適合度を求められる。

文書 ID タイプは単語ベクトルをハッシュ空間に投影しているため、複数単語の検索問い合わせの処理が軽いという利点がある。しかし、LSI のベクトルを構築するコストや、文書を LSI で次元を落として抽象化することが適合度の計算に及ぼす影響が欠点として挙げられる。文書 ID タイプの検索手法を示したものが図 2.8 である。

**term-to-peer 索引タイプ** term-to-peer 索引タイプでは、ノードに対して、ノードの保持する文書中の単語の ID が割り当てられる。つまり、ノードには保持する文書に登場する単語の数だけ ID が割り当てられる。各ノードは保持する文書コレクションに含まれる単語それぞれについて、DHT 上でその単語を担当するノードを探し、ノードリストに自分のアドレスを登録する。検索の際には、クエリに含ま

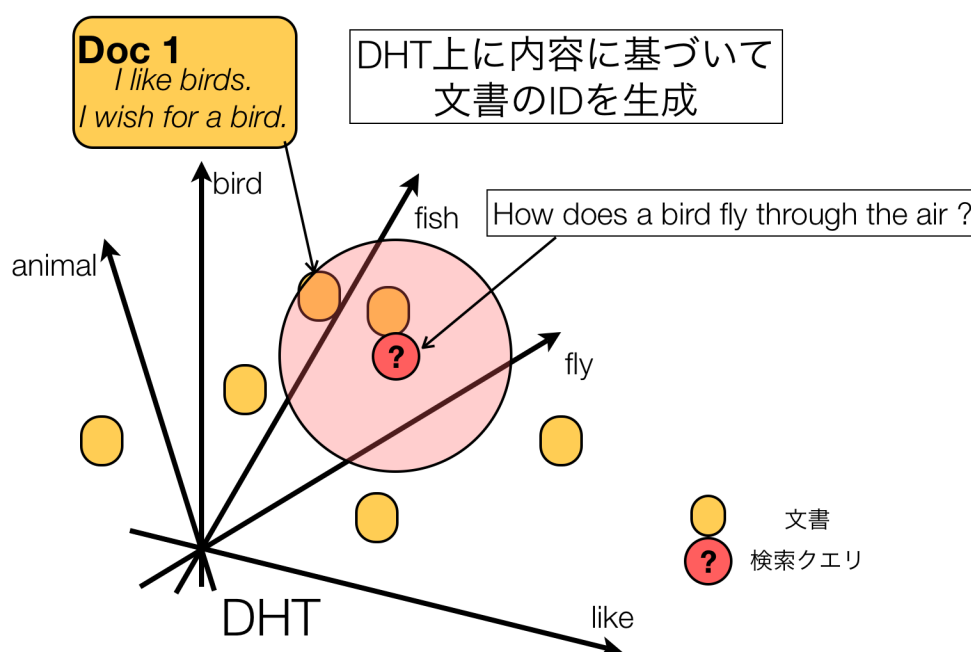


図 2.8: 文書 ID タイプ

れる単語のノードリストを参照し、該当するノードに接続することで関連する文書を集める。term-to-peer 索引は term-to-document 索引に比べて索引構築コストを抑えられる。しかし、索引のみから個々の文書を特定することができないという欠点がある。そのため、文書の問い合わせとの適合度を算出するには、検索問い合わせに関わる文書を保持するノードすべてに接続する必要がある。この接続コストを減らす目的で、ノードの評価を索引から行う試みがされている。

MINERVA では、検索問い合わせに適合する文書を保持するノードを選択するには、ノードがローカルに持つ文書に出現する特徴的な単語の割合を考慮した CORI2[Cal00] と呼ばれる手法が適していると評価した [BMT<sup>+</sup>05]。他にも、ノードの持つ文書コレクションの重複を考慮する手法 [BMT06; MBN06] や、適合度の計算の正確さを向上するために総文書数を Hash Sketches で推定する手法 [BMTW06] が提案されている。

term-to-peer 索引タイプは検索問い合わせに適合する文書を持つノードを高速に見つけられる利点がある。しかし、大規模な P2P ネットワークで多くの文書が重複して保持されている環境では、検索が非効率となってしまうことが欠点である。term-to-peer 索引タイプの検索手法を示したものが図 2.9 である。

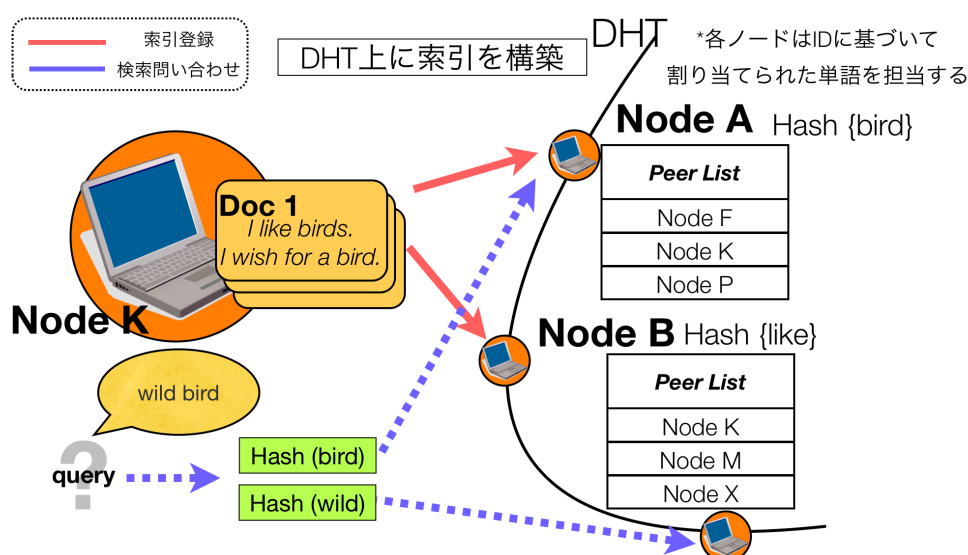


図 2.9: term-to-peer 索引タイプ

**term-to-document 索引タイプ** term-to-document 索引タイプでは、文書に対して、文書中に登場する単語の ID が割り当てられる。各ノードは保持する文書中の単語それぞれについて、DHT 上でその単語を担当するノードを探し、単語の索引に文書情報を登録する。検索の際には、クエリに含まれる単語の文書リストを参照し、適合度の高い文書を保持するノードのアドレスを取得し、それら文書を収集する。つまり、集中型検索における転置索引を P2P 環境で実現できる。しかし、索引が大きくなりやすく、索引参照に伴う負荷が大きくなってしまいう欠点がある。また、term-to-document 索引でも当てはまるが、索引構築のオーバーヘッドも大きいという欠点がある。

索引参照負荷については、以下のような取り組みがなされている。ODISSEA では、閾値の設定などを考慮した top-k 検索のためのアルゴリズムを提案した [SMW<sup>+</sup>03]。検索結果の上位に位置する文書を推定し、文書リストを絞り込むことで、索引参照におけるトラフィックを削減できる。2 つ以上の索引を参照するコストを削減しているのが Reynolds らの研究である [RV03]。Reynolds らは、Bloom filter [Blo70] とキャッシュを組み合わせることで複数単語を含む検索問い合わせの処理を効率化した。以上の 2 つは索引参照の過程の効率化を目指したものであるが、一方、Proof や HDKs は索引に登録する情報を工夫することで解決を図っている。Proof では、単語の索引に文書を登録する際に、文書情報を含ませた Bloom filter [Blo70] や文書の重みを同時に登録することで、検索結果から検索問い合わせとの適合度の低い

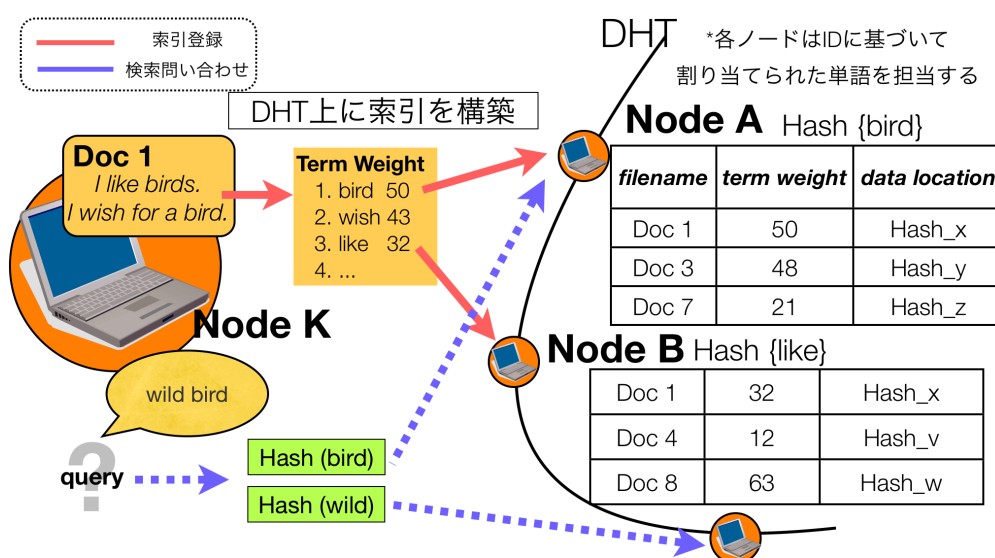


図 2.10: term-to-document 索引タイプ

文書を容易に除けるようにした [YH06] . HDKs では , 単語や複数単語の索引を構築して索引の種類を増やすことで , 索引への登録数の上限値を定めることで効率化した [PRL<sup>+</sup>07] .

索引構築のオーバーヘッドについては以下のような取り組みがなされている . 索引への登録数を大幅に削減することで解決を図っているのが Query-Driven Index である [SLZ<sup>+</sup>07] . Query-Driven Index では , 検索性能よりも索引構築コストを削減することを重視し , 検索結果のログを索引ノードに管理させるシステムを提案している . つまり , 既知の検索語を効率的に共有することを意図している . しかし , 過去に検索されていない単語についてはフラッディング手法で検索するため , 他の term-to-document 索引タイプの既存研究に比べて検索性能は低いと思われる .

term-to-document 索引タイプは集中型検索システムと同程度の検索精度を実現できる利点がある . しかし , 索引構築のオーバーヘッドを抑えて詳細で精度の高い索引を構築することが困難なことが欠点である . term-to-peer 索引タイプの検索手法を示したものが図 2.10 である .

## 2.2.4 P2P 環境における検索手法の比較

P2P 環境における検索手法を , 構築コスト , 検索性能の 2 つの観点から比較する .

検索システムの構築コストの観点から見ると , Unstructured P2P を用いた手法が優れている . 特に事前に索引やノードの保持する文書情報を共有しないクラスタ形

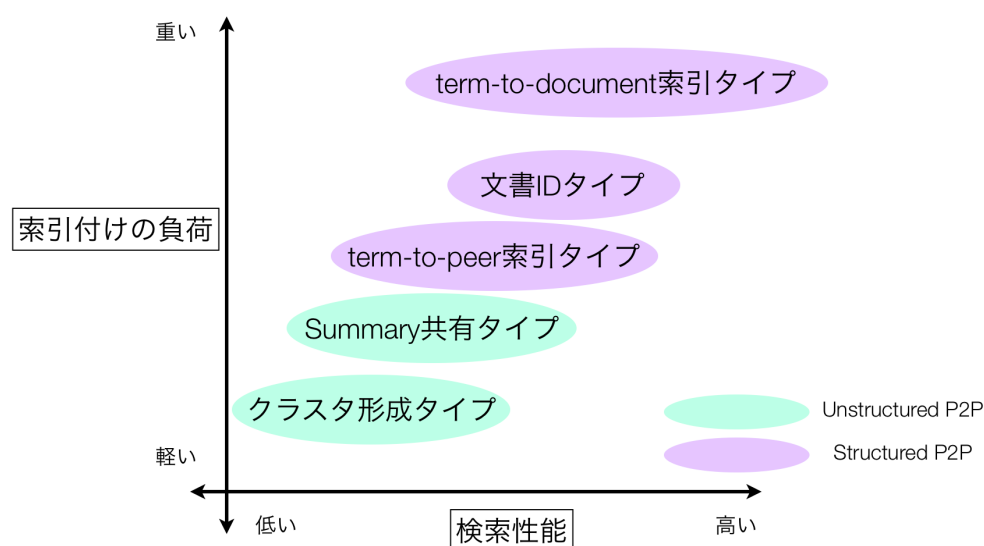


図 2.11: P2P 環境における索引付け手法の比較

成タイプが優れている。構築コストの低い Unstructured P2P に対して、Structured P2P では DHT による隣接ノードの制約がありノードの参加や離脱のたびに一定の通信が必要となってしまうため、構築コストが大きくなってしまう。Structured P2P における検索手法の中では term-to-document 索引タイプが、文書中のすべての単語の索引に登録する必要があるため、構築コストが最も大きい。

検索性能の観点から見ると、Structured P2P の term-to-document 索引タイプが最も優れている。term-to-document 索引タイプでは集中型と同様の詳細な索引を構築でき、さらにそれらを高速に参照できる。再現率という観点で見ると、大規模なネットワークでも、文書単位で索引に登録する term-to-document 索引タイプと文書 ID タイプは漏れなく適合文書を探すことが出来る。また、term-to-peer 索引タイプや Summary 共有タイプはトラフィックは生じてしまうが時間をかければ適合する文書をすべて探せられる。しかし、Unstructured P2P のクラスタ形成タイプは索引を構築しないため、再現率を高めることは困難である。

以上をまとめたものが図 2.11 である。

## 2.3 P2P 環境における負荷分散手法

P2P 環境における負荷分散手法について説明する。

Unstructured P2P ではデータと配置場所に制約がないため、各ノードは負荷に基づいて自由に保持するデータを調整できる。また、ニーズの高いデータほど複製がネットワーク内に配置することで、データへの問い合わせを効率良く分散している [gnu; CSWH01]。

一方、Structured P2P ではデータの管理がノードの ID によって決められる。そのため、ノードの ID がうまく分散されていない環境では一部のノードが担当するデータが膨大になるため、負荷が集中してしまう。このノードの ID は、多くの DHT ではハッシュ関数によって割り当てられ、確率論的に分散する設計になっている。既存研究ではノードの ID をさらに均一に分散させるため、以下に述べるような手法を提案している。

Chord ではノードに対して 1 つの ID を付与するのではなく、ネットワーク内に複数の Virtual Server を生成して、各 Virtual Server に ID を付与する [SMK<sup>+</sup>01]。つまり、ノードに対して複数の ID を付与する。これにより、各ノードの担当するデータ量のばらつきを相殺している。しかし、トポロジー維持コストの増大を招くことが欠点として挙げられる。

そこで、ノードの ID 付与をハッシュのみに頼るのではなく、一定の制約を加えることで改善した研究が提案されている。Naor らは、ランダムにノードを選び、そのノードが管理するハッシュ空間を新たに加わったノードと 2 分割するような ID を付与する手法を提案した [NW03]。既存 ID の管理していたハッシュ空間を均等に 2 分割していくことで、ID ごとのばらつきを軽減している。この手法では ID の分割対象となるノードはランダムに 1 つ選ばれるため、負荷分散の効果が小さい場合もある。そこで、Manku[Man04] は、ランダムに選んだノード周辺で最も管理するハッシュ空間が広いノードを選ぶようにし、負荷分散効果のばらつきを軽減する改善をした。さらに、それら 2 手法を組み合わせた手法を Kenthapadi らが提案している [KM05]。P2P 環境での負荷分散手法は、ID の付与手法だけでなくノードの近接性を考慮した手法なども提案されている [ZH05]。

既存の負荷分散手法はいずれも、データに対して 1 つの ID が割り当てられていて、それが膨大にネットワーク上に存在する環境を想定している。そのような環境では、データ数よりもノード数の方が少なく、ノードの ID の分布が負荷分散に与える影響が大きい。そのため、既存研究ではノードに割り当てる ID を分散させ

ることを主な研究課題にしている．

これら既存手法を Structured P2P で構築する P2P 情報検索に適用することは難しい．P2P 情報検索ではデータ(文書)に対して割り当てられる ID(単語)が複数ある．さらに，文書に割り当てられる ID は一部が頻繁に重複し，その重複分布は単語の出現確率と同様のものとなる．そのような環境では，データの ID によってノードに与える負荷が大きく異なるため，ノードの ID よりもデータの ID の分布が負荷分散に与える影響が大きい．そのため，ノードの ID を単純に分散させるだけでは，P2P 情報検索における負荷をノードに分散することは困難である．

## 2.4 P2P 環境におけるデータ配置手法

P2P 環境におけるデータ配置手法について述べる。

データの収集効率を目的としたデータ配置手法の多くは、データの複製生成と配置場所を工夫している。Unstructured P2P ではデータへのニーズに基づいて複製が配置され、負荷分散だけでなくデータの収集効率にも貢献している。Gnutella ではデータを要求したノードに複製が配置されるため、人気の高いデータほど検索で見つけやすい [gnu]。Freenet では問い合わせの経路上のノードに問い合わせを満たす文書のデータの複製が配置される [CSWH01]。その結果、文書のデータはその文書を要求する問い合わせのハッシュ値に近いノードに配置されやすくなる。つまり、Freenet はデータの内容に基づいた配置となり、検索の際に適合データを効率良く収集できる。しかし、Gnutella も Freenet も出現頻度の低い問い合わせに対しては、膨大な手間をかけて文書を収集することになる。

一方、Structured P2P ではデータはデータの ID に基づいて配置場所が決定される。そのため、単語ごとに索引を作成する P2P 情報検索手法ではいずれも、文書のデータの配置法は全ての文書が単語と無関係なノードに割り当てられている。つまり、索引を参照することでノードを絞り込むことはできても、ユーザは問い合わせに対する適合度の大小に関わらず同じだけの手間をかけて文書を取得しなければならない。これに対して、文書に割り当てる ID を単語ベクトルから生成する pSearch では、検索問い合わせに適合する文書ほど収集しやすい配置となっている [TXD03]。pSearch ではデータの配置場所が索引の役割も担っている。

データ転送ノードの負荷軽減を目的としたデータ配置手法として、SCOPE が挙げられる [CRWZ05]。SCOPE では RPT(Replica Partition Tree) を各データに生成し、複製配置場所の情報を分散して管理することで、効率的にデータ送受信の負荷を分散している。

データ収集効率と負荷分散を兼ね備えたアプローチとして、Network Coding が挙げられる [GR05]。データを Erasure 符号で分割することで、分割データをいくつか損失してもデータを取得できるようにしている。しかし、Network Coding は分割データから元のデータに復号する手間がかかる欠点がある。

## 2.5 既存研究の問題点と本研究での取り組み

筆者は、P2P 環境で大規模検索システムを容易に構築することを目指している。分散環境で集中型に比べて検索性能を落とすことなく精度の高い検索を実現するには、第 2.2.4 節で述べたように term-to-document 索引タイプが適切だと考えた。しかし、この term-to-document 索引タイプを大規模検索システムに適用するうえで、索引構築のオーバーヘッドや検索実行時のコストがボトルネックである。

索引構築における問題は 2 つあり、1 つは索引登録数が膨大なこと、もう 1 つは各索引における負荷が不均一なことである。前者は、term-to-document 索引タイプは文章中のすべての単語の索引に文書情報を登録する索引付け手法が要因であると考えられる。だが、検索精度を保つにはこの索引付け手法は避けられない。後者は、単語に割り当てられる ID が、ハッシュ関数を基に決められることが要因であると考えられる。一般に、単語の出現確率は Zipf の法則に基づくと言われている。そのため、頻出単語の索引を管理するノードには、索引登録のため膨大なアクセスを受けることになってしまう。第 2.3 節で述べたように、既存のノードの ID 割り当てを工夫した負荷分散手法での改善は困難である。

検索実行時のコストについては、既存研究では索引参照効率について主に取り組まれている。実際の検索過程を考えると、検索問い合わせに適合する文書リストを得るまでの効率だけでなく、適合文書を収集する効率も考慮する必要がある。第 2.4 節で述べたように、既存のデータ配置手法は、索引を作成する P2P 情報検索におけるデータの収集効率について考慮されていない。

そこで筆者は、2 つの研究に取り組むことにした。1 つは、各ノードにおける索引への被アクセス数を考慮した負荷分散のための索引構造の研究である。索引において単語を表す ID の割り当てを単語の出現確率を用いて工夫する。これにより索引構築の際に一部のノードへのアクセスの集中を防ぎ、索引登録負荷を分散することを目的とする。もう 1 つは、検索問い合わせに適合する文書データを効率良く収集できるようなデータ配置手法の研究である。これにより検索実行の際に高速に適合文書を収集し、検索応答時間を短縮することを目的とする。これら 2 つの研究を表したものが図 1.1 である。本稿では第 3 章にて索引構造、第 4 章にてデータ配置手法について説明する。

## 第 3 章

### **Huffman-DHT:**

### 単語の出現確率に基づいた索引構造

## 3.1 Huffman-DHT の概要

P2P ネットワークで情報検索システムを構築するとき、索引は分散して構築・管理される。この索引を構築したり管理する処理は非常に重い。例えば、P2P 情報検索システムにおいて新たな文書を索引に登録する場合、文書に含まれる各単語について、その単語の索引を管理するノードに接続する必要がある。また、検索を実行する場合も、検索問い合わせに含まれる各単語について、その索引を管理するノードに接続する必要がある。このような索引に関わるオーバーヘッドを抑える目的で、関連研究の多くは DHT を用いている。DHT 上に索引を構築することで、 $n$  ノードで構成されている P2P ネットワークでは  $O(\log n)$  で目的のノードへアクセスできる。しかしながら、依然として索引構築のオーバーヘッドは大きく、P2P IR におけるボトルネックである。なぜなら、DHT では単語の出現確率に関係なく、ハッシュ関数を用いて単語に ID が割り当てられ、この ID をもとに単語の索引を管理するノードが決まるからである。そのため、登録頻度の高い単語を管理するノードへアクセスが集中し、ホットスポットと呼ばれる現象が起きてしまう。また、単語の索引を管理するノードにアクセスするまでのコストは、登録頻度の高い単語もそうでない単語も同一である。

そこで、筆者は P2P 情報検索のための新たなノードアクセス手法である Huffman-DHT を提案する。Huffman-DHT は索引登録時のホットスポットを解消し、さらに索引構築に伴う探索コストを低減することを目的とする。これを実現するため、Zipf の法則の通りに登録頻度の高い一部の単語の索引が索引構築の大部分を占めている特徴を利用し、Huffman-DHT では出現確率の高い単語に対して ID 空間で広い領域を割り当てる。ID 空間の分割には、符号理論の Huffman 符号を採用した。Huffman-DHT を用いることで、索引構築の際の被アクセス数をノード間で負荷分散できる。さらに、通常の索引では単語の索引を保持するノードを探すのにその単語の出現確率に関わらず  $O(\log n)$  のホップ数必要となるところを、登録頻度の高い単語の索引ほど少ないホップ数で探せる。

本研究の貢献は、以下の 2 点である。

- Structured 型 P2P 情報検索における索引構築の処理を符号理論を用いて分散した。
- 登録頻度の高い単語の索引ほど短時間で登録できるように、出現確率の高い単語に対して ID 空間で広い領域を割り当てた。

本章の構成は以下の通りである．まず，第 3.2 節では Huffman-DHT のアーキテクチャについて述べる．第 3.3 節にて本手法の評価を行い，第 3.4 節でまとめと考察を行う．

## 3.2 Huffman-DHT のアーキテクチャ

本節では、Huffman-DHT の構造、ノードの管理する ID、ノードの探索、そして文書の索引への登録方法について説明する。

### 3.2.1 Huffman-DHT の構造

Huffman-DHT では、すべてのノードと単語に ID が割り当てられる。Huffman-DHT で用いられる ID は  $b_1b_2\cdots b_l$  のビット列となっており、 $b_i$  ( $1 \leq i \leq l$ ) は 0 か 1 である。ノードの ID はハッシュ関数を用いて割り当てられ、単語の ID は後に述べる Huffman-DHT の符号木を用いて割り当てられる。ここで、ノード  $p$  と  $q$  の距離は、 $id(\cdot)$  をノードの ID と仮定すると、 $|id(p) - id(q)|$  で表すことが出来る。一般的な DHT と同様に、各ノードは ID を基に DHT における ID 空間が割り当てられ、また、他のノードへの経路表の一部を保持する。

単語の ID は Huffman-DHT の符号木を用いて決定される。文書を索引に登録する際に発生する単語の索引へのアクセス数はその単語の Document Frequency と同値であることから、Huffman-DHT の符号木は単語の出現確率を基につくられる。Huffman-DHT で利用する単語の出現確率は、文書コレクションの一部の文書セットにおける DF 値から推定する。推定の手順は以下の通りである。 $df(t, S)$  を文書セット  $S$  における単語  $t$  の Document Frequency とすると、 $df(t, S)$  を降順に並べたときの上位  $k$  単語を選ぶ。そして、頻出単語群  $T$  に含まれる単語  $t$  の出現確率  $p(t)$  を、

$$p(t) \equiv \frac{df(t, S)}{\sum_{t \in T} df(t, S)} , \quad (3.1)$$

と推定する。これら頻出単語の出現確率の推定値を用いて、Huffman 符号のアルゴリズムに基づいて、符号木を構築する。出現頻度の上位  $k$  単語に含まれなかった単語については、その単語のハッシュ値に最も近いハッシュ値の頻出単語と同一の ID が割り当てられる。図 3.1 は符号木の構築方法を示す。図でも示すように Huffman-DHT の符号木は 2 分木である。

本研究で頻出単語のみを符号化する理由は以下の通りである：

- すべてのノードで共有する必要がある符号木のサイズを減らすため。
- 頻出単語に対して効果的な符号化であるため。

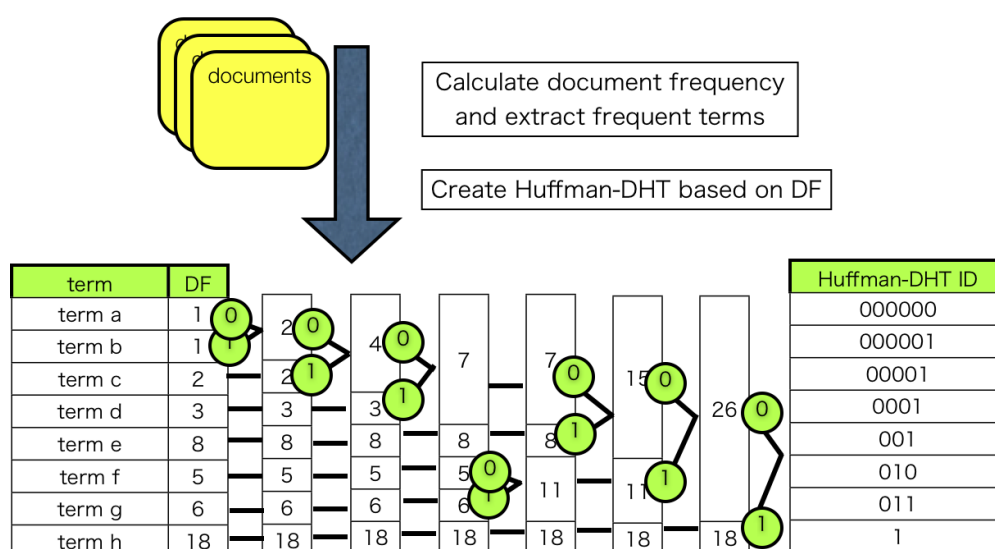


図 3.1: Huffman-DHT の符号木の構築方法

- 文書コレクションの一部から単語の出現分布を推定するため。

### 3.2.2 ノードの管理する ID

頻出単語  $t$  に対して割り当てられる ID を  $c_1 \cdots c_m$  と仮定する。単語の ID の長さ  $m$  がノードの ID のビット長  $l$  よりも長い場合、単語の ID は  $l$  ビットに切り詰められ、単語の索引は短縮された ID の  $c_1 \cdots c_l$  に最も近いノードに割り当てられる。一方、単語の ID の長さ  $m$  がノードの ID のビット長  $l$  よりも短い場合、単語の索引は  $\underbrace{c_1 \cdots c_m 0 \cdots 0}_l$  から  $\underbrace{c_1 \cdots c_m 1 \cdots 1}_l$  の範囲の ID に該当するノードに割り当てられる。この範囲にノードが存在しない場合は、ID に最も近いノードに割り当てられる。Huffman-DHT では頻出単語ほど短い ID が割り当てられる。それゆえ、頻出単語の管理に多くのノードが割り当てられることになる。図 3.2 は図 3.1 に基づいて頻出単語が ID 空間に割り当てられる様子を示している。図 3.3 はノードの管理する単語を示している。

### 3.2.3 ノードの探索

Huffman-DHT では、頻出単語のリストと第 3.2.1 節で述べた符号木を全ノードが保持する。本節では、P2P IR で検索を行う際に、検索問い合わせに含まれる単語  $t$  を管理するノードを検索実行ノードが探す手順を説明する。

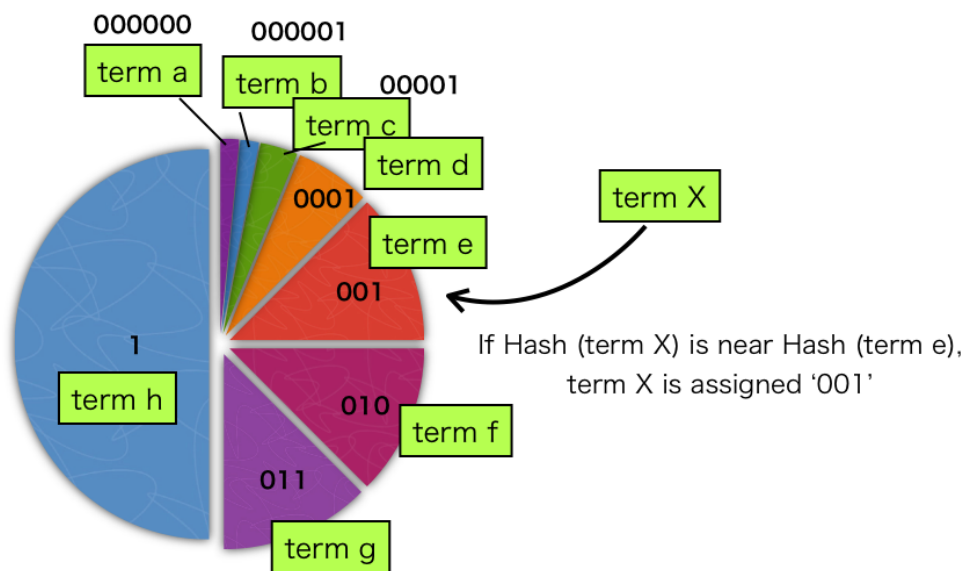


図 3.2: Huffman-DHT における単語 ID の割り当て方法

1. 単語  $t$  の ID を求める .
  - (a) 単語  $t$  が頻出単語のリストに含まれる場合 , 検索実行ノードは単語  $t$  の ID を符号木から導く .
  - (b) 単語  $t$  が頻出単語のリストに含まれない場合 , 検索実行ノードは単語  $t$  のハッシュ値を求め , そのハッシュ値に最も近い頻出単語を選ぶ . そして , その頻出単語の ID を単語  $t$  に割り当てる .
2. 検索実行ノードが単語  $t$  の ID に該当するノードが経路表にないか確認する . 経路表にある場合はそこに記されているアドレスを参照する .
3. 経路表にない場合は , 単語  $t$  の ID に最も近い ID のノードに接続し , そのノードの経路表を参照する . これを該当するノードを見つけるまで繰り返す .

P2P ネットワーク内にノードが十分に存在するとき , 頻出単語には多くのノードが割り当てられているため , 少ないホップ数で索引管理ノードへたどり着ける .

### 3.2.4 文書の索引への登録

Huffman-DHT における文書の索引への登録方法は以下の通りである .

1. 文書を単語に分割し , ストップワードの除去とステミングを行う .

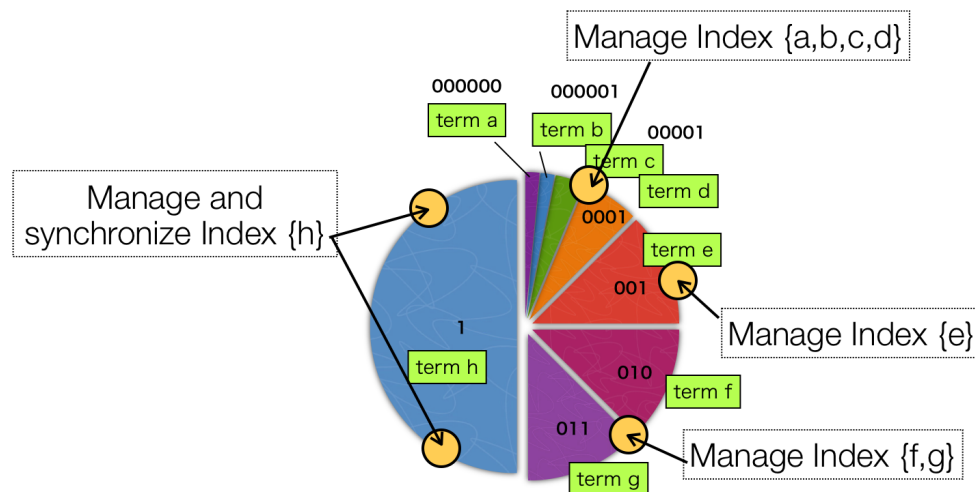


図 3.3: Huffman-DHT におけるノードの管理する ID

2. 文書中の単語  $t$  について

- (a) 第 3.2.3 節で述べたように，単語  $t$  の索引を管理するノードを探索する．
- (b) 単語  $t$  の索引に文書の情報を書き込む．

Huffman-DHT では単語の索引を管理するノードが複数ある場合は，それらのうちの 1 つのノードの索引を更新する．そのため，索引に何か情報を追加したノードは，他の索引ノードへ更新情報を送信する必要がある．索引管理ノードは連続した ID で構成されるため，あまり負荷をかけずに効率良く更新できる．

### 3.3 Huffman-DHT の評価

Huffman-DHT は、出現頻度の偏った単語をバランス良くノードで管理することを目的として、静的な ID を単語に割り当てる。筆者は Huffman-DHT を評価するためシミュレーションを用いた実験を行った。実験では2種類の文書コレクションを用いた。1つは、TREC[tre] の Text Research Collection Volume 3 (Tipster 3) を用いた。The San Jose Mercury News (1991) , the Associated Press (1990) , U.S. Patents (1983-1991) , そして Information from the Computer Select disks (1991, 1992) copyrighted by Ziff-Davis の4種類からなる合計 33.6 万の文書データである。英単語のステミングには Porter stemmer[por] を用いた。もう1つの文書コレクションは NTCIR-4 WEB[ntc] の NW100G-01 である。JP ドメインの Web で提供される文書からなる合計 1100 万の文書である。形態素解析には MeCab[mec] を用いた。

#### 3.3.1 単語の出現確率分布の推定精度

文書を索引に登録する際に発生する単語の索引へのアクセス数はその単語の Document Frequency と同値である。索引へのアクセスを分散させるということはつまり、各ノードに割り当てる単語を DF 値が分散するように設定することである。そのためには、DF 値の分布を推定することが必要となる。

まず、文書コレクションにおける Document Frequency の分布がどのようなになっているか実験を行った。図 3.4 は Tipster 3 と NTCIR4 Web コーパスにおける DF の分布である。Tipster 3 は 33.6 万の文書データである。817,897 種類の単語を含み、DF 値の合計は 53,445,728 である。一方、NW100G-01 は 16,396,001 種類の単語を含み、DF 値の合計は 1,838,394,409 である。2つのコーパスの言語やデータサイズは異なるが、同じような Zipf の法則に基づいた出現頻度の分布となっていることがわかる。このような分布において頻度の高い単語が全体の出現頻度で占める割合を示したものが図 3.5 である。Tipster3 では出現頻度の高い上位 4,580 単語 (全単語の 0.56%) が単語出現数の 80% を占めていることがわかった。これより、索引構築時にこれら一部の頻出単語を効率よく扱うことができれば、索引構築に必要なコストを抑えられると考えられる。

次に、文書コレクションにおける単語の出現確率の推定についての実験を行った。Huffman-DHT を P2P IR システムで構築するためには、文書における単語の出現確率を全文書が索引に登録されるより前に推定する必要がある。つまり、文書

における単語の出現確率を文書コレクションの一部から推定する必要がある．筆者は Tipster 3 からランダムに選んだ文書を用いて，単語の出現確率をどの程度推定できるか実験を行った．実験では，ランダムに選ぶ文書数を変化させたとき，選んだ文書セットと文書コレクションとの DF 値の分布がどの程度類似しているかを Kullback-Leibler 情報量で評価した．まず，ランダムに選んだ文書セットから導く出現確率に対して，単語が含まれていない場合も Kullback-Leibler 情報量を計算できるように，Laplace smoothing でスムージング処理した．通常ならランダムに選んだ文書セットにおける単語の文書への出現確率  $P_{base}(t, S)$  は，

$$P_{base}(t, S) = \frac{df(t, S)}{\sum_{t' \in T} df(t', S)}, \quad (3.2)$$

となるが，Laplace smoothing を適用することで，

$$P_{Ls}(t, S) = \frac{df(t, S) + 1}{\sum_{t' \in T} df(t', S) + 1}, \quad (3.3)$$

となる．式で用いられている  $T$  は文書コレクション中に登場する単語， $S$  はランダムに選んだ文書セットである，この出現確率を用いて，Kullback-Leibler 情報量  $KL$  は，

$$KL = \sum_{t \in T} P(t, D) \cdot \log \frac{P(t, D)}{P_{Ls}(t, S)}, \quad (3.4)$$

で求められる．式で用いられている  $D$  は文書コレクションである，実験結果から，Kullback-Leibler 情報量の値が少ない文書数で収束していることがわかった．つまり，一定数以上の文書が手元があれば文書コレクションにおける単語の出現確率を十分に推定できることがわかった．この結果より，同種の文書が索引に追加される場合は，Huffman-DHT の符号木を頻繁に更新する必要がないことがわかった．

### 3.3.2 索引付けの際のホップ数

索引付けの際に必要なホップ数について，理論値とシミュレーションを用いて実験を行った．

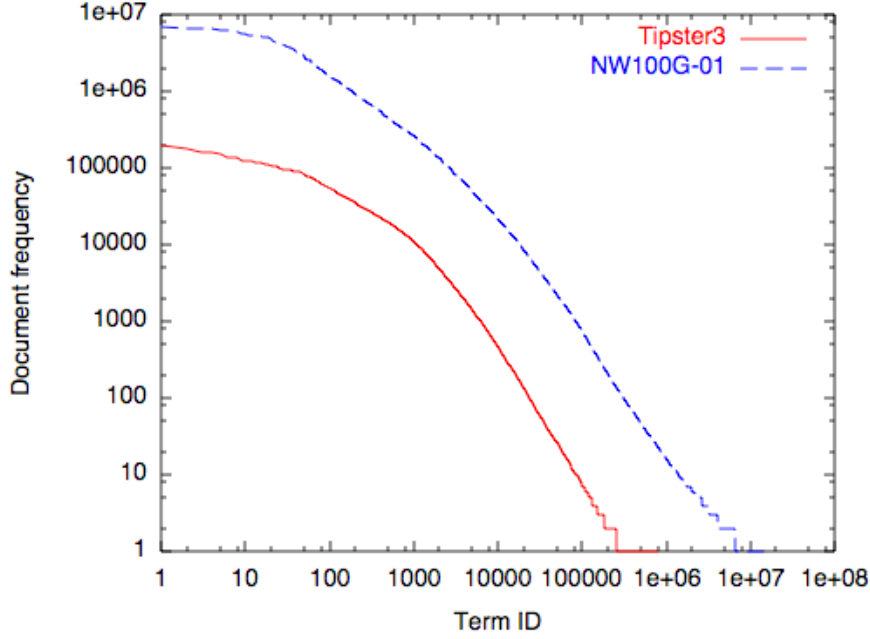


図 3.4: 文書コレクションにおける Document Frequency の分布

#### 理論値による評価

既存の DHT を用いた P2P IR における索引時に発生するホップ数の最大値は単語にかかわらず一定である。P2P ネットワーク上に存在するノード数を  $N_{node}$  とすると、単語を管理するノードにたどり着くまでのホップ数は、 $O(\log N_{node})$  となる。各ノードの経路表に登録されたノード数に依存した変数  $m$  を仮定すると、 $\frac{\log N_{node}}{m}$  となる。文書コレクションに出現する単語の種類を  $N_{term}$ 、DF 値の合計を  $N_{all}$ 、とすると、索引時の総ホップ数  $H_{baseline}$  は、

$$H_{baseline} = N_{all} \cdot \frac{\log N_{node}}{m}, \quad (3.5)$$

となる。

一方、Huffman-DHT では、単語の出現確率に基づいてツリー構造が構築される。単語  $t_k$  の文書への出現確率  $P_{t_k}$  は、Zipf の法則より、

$$P_{t_k} = \frac{\frac{1}{k^s}}{\int_{n=1}^{N_{term}} \frac{1}{n^s}}. \quad (3.6)$$

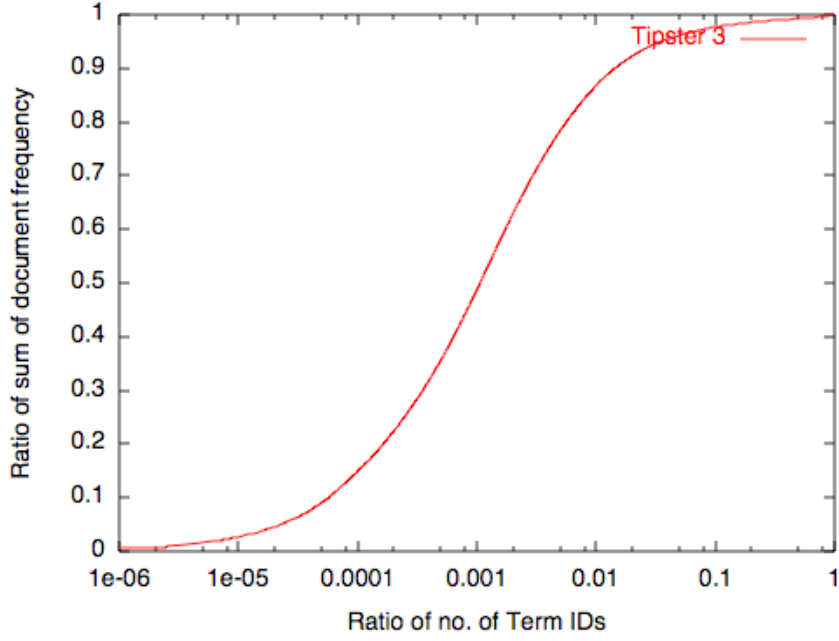


図 3.5: 頻度の高い単語の DF 値が全体で占める割合

単語  $t_k$  の DF 値  $N_{t_k}$  は ,

$$N_{t_k} = N_{all} \cdot \frac{\frac{1}{k^s}}{\int_{n=1}^{N_{term}} \frac{1}{n^s}}, \quad (3.7)$$

で表される .

Huffman-DHT で構築されるツリーにおける単語  $t_k$  の符号長  $l_{t_k}$  は ,  $-\log P_{t_k}$  に近い値となる . 実際は , 符号長が整数になるように近似されるため ,  $-\log P_{t_k}$  より若干大きい値となる . 式で表すと ,

$$\begin{aligned} l_{t_k} &= -\log P_{t_k} + \alpha \quad (\alpha \geq 0) \\ &= -\log \frac{\frac{1}{k^s}}{\int_{n=1}^{N_{term}} \frac{1}{n^s}} + \alpha. \end{aligned} \quad (3.8)$$

$\int_{n=1}^{N_{term}} \frac{1}{n^s}$  を  $\beta(s)$  とおくと ,

$$\begin{aligned} l_{t_k} &= -\log \frac{\frac{1}{k^s}}{\beta(s)} + \alpha \\ &= s \cdot \log k + \log \beta(s) + \alpha, \end{aligned} \quad (3.9)$$

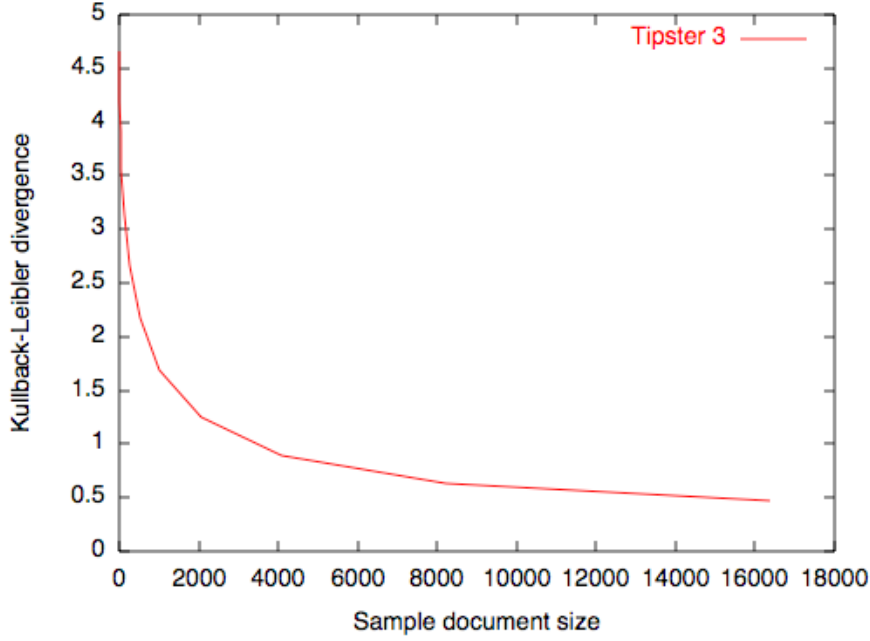


図 3.6: 出現確率の推定値と実際の値との類似度

となる．

DHT における探索に必要なホップ数は，探索対象とするノード群の ID の桁数（符号長），つまり，ツリーの深さに依存する．既存手法のホップ数と同様に，各ノードの経路表に登録されたノード数に依存した  $m$  を用いて，符号長を  $\frac{1}{m}$  倍した値がホップ数となる．以上のことから，単語  $t_k$  を管理するノードにたどり着くまでのホップ数  $h_{t_k, Huffman-DHT}$  は，

$$h_{t_k, Huffman-DHT} = \min \left[ \frac{s \cdot \log k + \log \beta(s) + \alpha}{m}, \frac{\log N_{node}}{m} \right], \quad (3.10)$$

となる．境界は，

$$\begin{aligned}
 \frac{s \cdot \log k + \log \beta(s) + \alpha}{m} &= \frac{\log N_{node}}{m} \\
 s \cdot \log k &= \log N_{node} - \log \beta(s) - \alpha \\
 s \cdot \log k &= \log \frac{N_{node}}{\beta(s) \cdot e^\alpha} \\
 \log k &= \frac{1}{s} \cdot \log \frac{N_{node}}{\beta(s) \cdot e^\alpha} \\
 k &= \left( \frac{N_{node}}{\beta(s) \cdot e^\alpha} \right)^{\frac{1}{s}}, \tag{3.11}
 \end{aligned}$$

のときである．これより，単語  $t_k$  を管理するノードにたどり着くまでのホップ数は，

$$h_{t_k, Huffman-DHT} = \begin{cases} \frac{s \cdot \log k + \log \beta(s) + \alpha}{m}, & k < \left( \frac{N_{node}}{\beta(s) \cdot e^\alpha} \right)^{\frac{1}{s}} \cap 1 \leq k \leq N_{term} \\ \frac{\log N_{node}}{m}, & other \end{cases}$$

と表せる．

$H_{Huffman-DHT}$  と既存手法のホップ数の関係は，

$$\begin{aligned}
 H_{Huffman-DHT} &= \int_1^{N_{term}} N_{t_k} \cdot h_{t_k, Huffman-DHT} dk \\
 &= \int_1^{N_{term}} N_{t_k} \cdot \min \left[ \frac{s \cdot \log k + \log \beta(s) + \alpha}{m}, \frac{\log N_{node}}{m} \right] dk \\
 &\leq \int_1^{N_{term}} N_{t_k} \cdot \frac{\log N_{node}}{m} dk = H_{baseline}, \tag{3.12}
 \end{aligned}$$

から， $H_{Huffman-DHT} \leq H_{baseline}$  が常に成り立つことがわかる．

索引時の総ホップ数  $H_{Huffman-DHT}$  は，

$$\begin{aligned}
 \left( \frac{N_{node}}{\beta(s) \cdot e^\alpha} \right)^{\frac{1}{s}} &\leq N_{term} \\
 N_{node} &< \frac{N_{term}^s}{\beta(s) \cdot e^\alpha},
 \end{aligned}$$

のとき ,

$$\begin{aligned}
H_{Huffman-DHT} &= \int_1^{N_{term}} N_{t_k} \cdot h_{t_k, Huffman-DHT} dk \\
&= \int_1^{N_{term}} \frac{N_{all}}{\beta(s)} \cdot \frac{1}{k^s} \cdot h_{t_k, Huffman-DHT} dk \\
&= \int_1^{\left(\frac{N_{node}}{\beta(s) \cdot e^\alpha}\right)^{\frac{1}{s}}} \frac{N_{all}}{\beta(s)} \cdot \frac{s \cdot \log k + \log \beta(s) + \alpha}{m \cdot k^s} dk \\
&\quad + \int_{\left(\frac{N_{node}}{\beta(s) \cdot e^\alpha}\right)^{\frac{1}{s}}}^{N_{term}} \frac{N_{all}}{\beta(s)} \cdot \frac{\log N_{node}}{m \cdot k^s} dk \\
&= \int_1^{\left(\frac{N_{node}}{\beta(s) \cdot e^\alpha}\right)^{\frac{1}{s}}} \frac{N_{all}}{\beta(s)} \cdot \frac{s \cdot \log k + \log \beta(s) + \alpha}{m \cdot k^s} dk \\
&\quad + \frac{N_{all} \cdot \log N_{node}}{m} \cdot \left(1 - \frac{1}{\beta(s)} \int_1^{\left(\frac{N_{node}}{\beta(s) \cdot e^\alpha}\right)^{\frac{1}{s}}} \frac{1}{k^s} dk\right) \\
&= \frac{N_{all} \cdot s}{m \cdot \beta(s)} \cdot \int_1^{\left(\frac{N_{node}}{\beta(s) \cdot e^\alpha}\right)^{\frac{1}{s}}} \frac{\log k}{k^s} dk \\
&\quad + \left(\frac{N_{all}}{m \cdot \beta(s)} \cdot (\log \beta(s) + \alpha - \log N_{node})\right) \cdot \int_1^{\left(\frac{N_{node}}{\beta(s) \cdot e^\alpha}\right)^{\frac{1}{s}}} \frac{1}{k^s} dk \\
&\quad + \frac{N_{all} \cdot \log N_{node}}{m}.
\end{aligned}$$

ここで ,  $\left(\frac{N_{node}}{\beta(s) \cdot e^\alpha}\right)^{\frac{1}{s}}$  を  $\gamma(s)$  と置いたとき ,

$$\begin{aligned}
&= \frac{N_{all} \cdot s}{m \cdot \beta(s)} \cdot \int_1^{\gamma(s)} \frac{\log k}{k^s} dk \\
&\quad + \left(\frac{N_{all}}{m \cdot \beta(s)} \cdot (-s \log \gamma(s))\right) \cdot \int_1^{\gamma(s)} \frac{1}{k^s} dk \\
&\quad + \frac{N_{all} \cdot \log N_{node}}{m}.
\end{aligned} \tag{3.13}$$

$s \neq 1$  のとき ,

$$\begin{aligned}
 \int \frac{\log k}{k^s} dk &= \int \log k \cdot \left( \frac{k^{-s+1}}{-s+1} \right)' dk \\
 &= \frac{1}{-s+1} \cdot \log k \cdot k^{-s+1} - \int \frac{1}{k} \cdot \left( \frac{k^{-s+1}}{-s+1} \right) dk \\
 &= \frac{1}{-s+1} \cdot \log k \cdot k^{-s+1} - \frac{k^{-s+1}}{(-s+1)^2} + C, \quad (C : \text{積分定数}) \quad (3.14)
 \end{aligned}$$

$$\int \frac{1}{k^s} dk = \frac{1}{-s+1} \cdot k^{-s+1} + C, \quad (C : \text{積分定数}) \quad (3.15)$$

であるから , 式 (3.13) は  $s \neq 1$  のとき ,

$$\begin{aligned}
 H_{Huffman-DHT} &= \frac{N_{all} \cdot s}{m \cdot \beta(s)} \cdot \left( \frac{\log \gamma(s) \cdot \gamma^{-s+1}(s)}{-s+1} - \frac{\gamma^{-s+1}(s) - 1}{(-s+1)^2} \right) \\
 &\quad + \left( \frac{N_{all}}{m \cdot \beta(s)} \cdot (-s \log \gamma(s)) \right) \cdot \frac{\gamma^{-s+1}(s) - 1}{-s+1} \\
 &\quad + \frac{N_{all} \cdot \log N_{node}}{m} \\
 &= H_{baseline} + \frac{N_{all} \cdot s}{m \cdot \beta(s) \cdot (-s+1)^2} \cdot \left( (-s+1) \cdot \log \gamma(s) - \gamma^{-s+1}(s) + 1 \right). \quad (3.16)
 \end{aligned}$$

となる .

一方 , ノードが膨大に存在するとき , Huffman-DHT では同期ノードが単純に増大していくことになる . つまり ,

$$N_{node} \geq \frac{N_{term}^s}{\beta(s) \cdot e^\alpha} ,$$

の数のノードでは , ホップ数は

$$\begin{aligned}
 H_{Huffman-DHT} &= \int_1^{N_{term}} \frac{N_{all}}{\beta(s)} \cdot \frac{1}{k^s} \cdot h_{t_k, Huffman-DHT} dk \\
 &= \int_1^{N_{term}} \frac{N_{all}}{\beta(s)} \cdot \frac{s \cdot \log k + \log \beta(s) + \alpha}{m \cdot k^s} dk \\
 &= \frac{N_{all} \cdot s}{m \cdot \beta(s)} \cdot \int_1^{N_{term}} \frac{\log k}{k^s} dk + \frac{N_{all} \cdot (\log \beta(s) + \alpha)}{m \cdot \beta(s)} \cdot \int_1^{N_{term}} \frac{1}{k^s} dk \\
 &= \frac{N_{all} \cdot s}{m \cdot \beta(s)} \cdot \left( \frac{1}{-s+1} \cdot \log N_{term} \cdot N_{term}^{-s+1} - \frac{N_{term}^{-s+1} - 1}{(-s+1)^2} \right) \\
 &\quad + \frac{N_{all} \cdot (\log \beta(s) + \alpha)}{m \cdot \beta(s)} \cdot \left( \frac{1}{-s+1} \cdot (N_{term}^{-s+1} - 1) \right), \quad (3.17)
 \end{aligned}$$

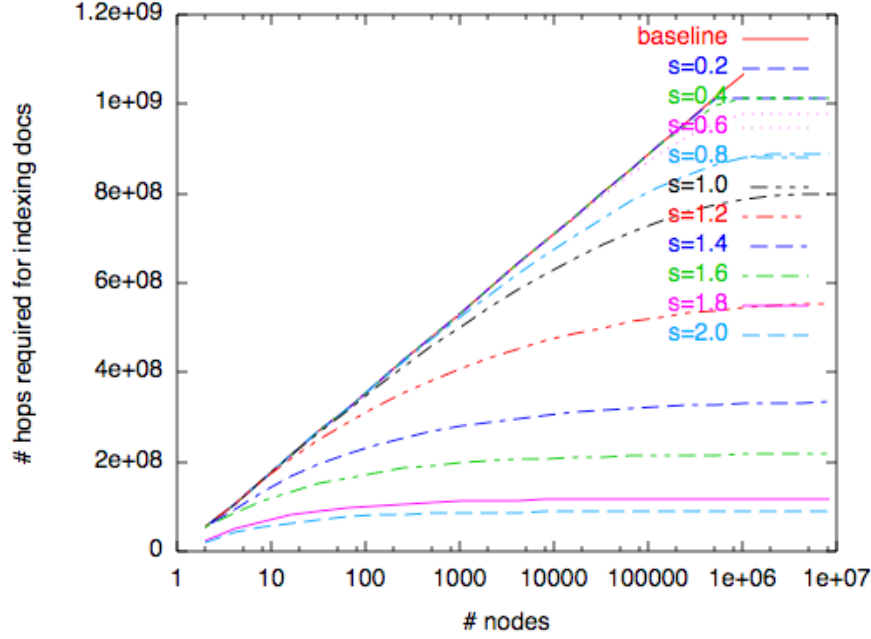


図 3.7: 索引付けの際のホップ数 (理論値)

という値で一定となる．この値は，ノードの数を増大させた場合の収束値ととらえられる．

以上の計算値を基に，Tipster3 における値  $N_{term} = 817,897$ ， $N_{all} = 53,445,728$  を代入して  $s$  の値を変化させたときの索引構築時のホップ数を比較したものが図 3.7 である．Zipf の法則における変数  $s$  の値が大きいほど，Huffman-DHT の効率が向上していることがわかった．つまり，Huffman-DHT では登録頻度の高い単語が偏る文書コレクションほど索引構築の効率が増すと言える．

#### シミュレーションによる評価

理論値による評価の際には，単語の出現確率が Zipf の法則に従うことを仮定した．本評価では，実際の文書コレクションを索引付けするときに必要なホップ数をシミュレーションで求めた．文書コレクションには Tipster 3 を用いた．本実験で用いる Huffman-DHT は Tipster 3 における 4,580 の頻出単語を基に構築した．これら単語は全単語の DF 値の合計の 80% を占める．Huffman-DHT の符号木の深さは 8 から 15 となった．P2P IR において索引付けに必要なホップ数を Huffman-DHT

と通常の DHT とで比較したものが図 3.8 である。

図中の赤線は単語の出現確率を考慮しない通常の DHT を用いたときの必要なホップ数である。

緑線は最も条件の悪い場合の必要なホップ数  $H_{worst}$  を示しており、

$$H_{baseline} = \sum_{t \in T} df(t, D) \times \log n, \quad (3.18)$$

で求めた。ここで、式で用いている  $T$  は文書コレクション中に登場する単語、 $D$  は文書コレクション、 $n$  はノードの数である。この式は、文書を 1 つずつ索引に登録することを想定し、文書中の各単語を管理するノードを  $\log n$  ホップで探す場合に必要なホップ数を表す。

青線は筆者の提案する Huffman-DHT を用いて索引付けしたときの必要なホップ数である。

図からわかるように、通常の DHT(赤線)は最も悪い条件(緑線)よりもホップ数は少ない。これは、探索対象のノードがユーザのノードに近い場合は少ないホップ数で探索が終了するためである。だが、通常の DHT は最も悪い条件の結果と比例関係になっている。一方、提案手法である Huffman-DHT(青線)は索引付けの際に必要なホップ数を削減できていることがわかる。図から 1,000 ノードよりも大きな P2P システムにおいて Huffman-DHT が既存手法よりも効率よくなることがわかった。これより、Huffman-DHT は通常の DHT よりもスケーラブルな設計であると言える。

### 3.3.3 索引登録時の被アクセス数の分散度合い

通常の DHT を P2P IR で用いた場合、文書に頻出する単語を管理するノードに対して索引登録のアクセスが集中する。その結果、ホットスポットと呼ばれる現象が起きてしまう。一方、Huffman-DHT では単語の出現確率に基づいた ID 割り当てによりノードの管理する単語の出現数がおよそ均一になるため、索引へのアクセスがノードに効率よく分散される。

筆者は P2P IR における索引登録の被アクセス数について実験を行った。実験では、文書コレクションを索引登録する際に生じるノードへの被アクセス数について、ハッシュ関数を基に単語をノードに割り振る通常の DHT と Huffman-DHT とで比較を行った。文書コレクションには Tipster 3 を用いた。Huffman-DHT は第 3.3.2

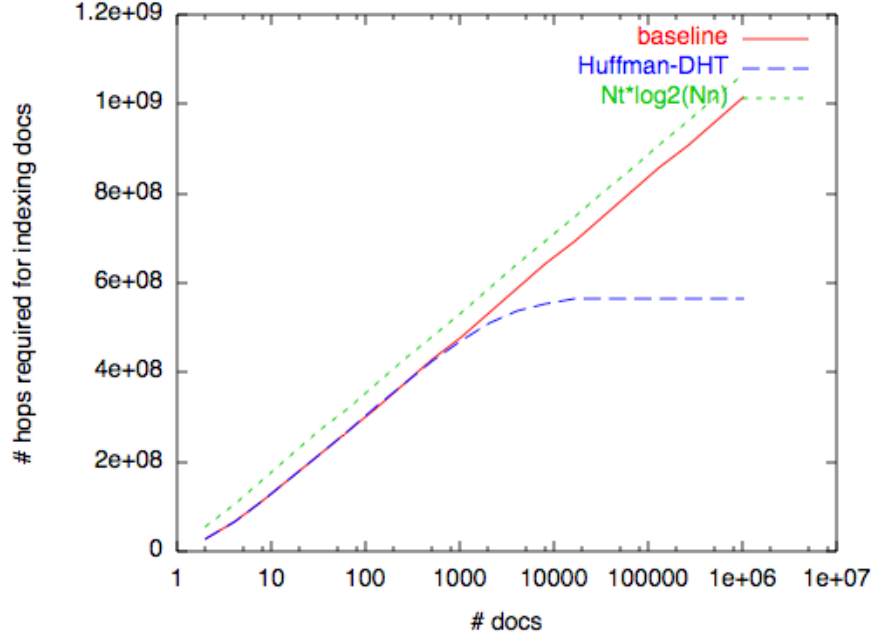


図 3.8: 索引付けの際のホップ数 (シミュレーション)

節と同様に 4,580 の頻出単語を基に構築した．被アクセス数の分散度合いを示す指標として，被アクセス数の最大値と最小値の比  $V_{access}$  の

$$V_{access} = \frac{\max_{p \in N} a(p)}{\min_{p \in N} a(p)}, \quad (3.19)$$

を評価に用いた．ここで， $N$  は P2P IR を構成するノードの集合， $a(p)$  はノード  $p$  における被アクセス数を表す．ノード数を変化させたときの被アクセス数の分散度合いを示したものが図 3.9 である．赤線は通常の DHT を用いたときであり，青線は Huffman-DHT を用いたときを表している．実験結果より，100 ノードよりも大きな P2P システムにおいて Huffman-DHT が既存手法よりも効率よくなることがわかった．これより，Huffman-DHT によって同じ ID が割り当てられたノードにおいて索引の更新を頻繁に同期しても，被アクセス数の分散度合いを維持できることがわかった．

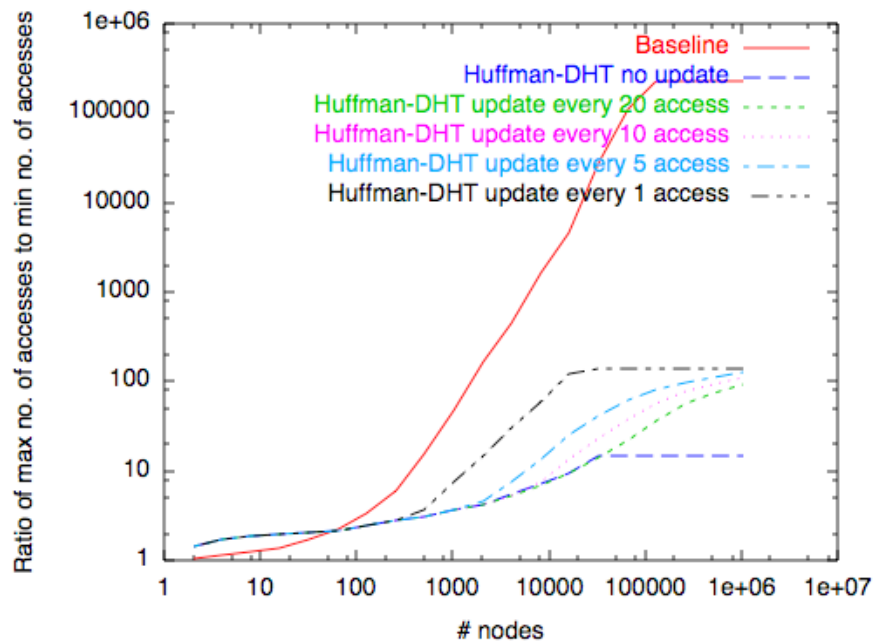


図 3.9: 索引登録時の被アクセス数の分散度合い

### 3.3.4 索引の同期が必要なノード数

Huffman-DHT では同じ ID が割り当てられたノードで索引の更新を同期する必要がある．第 3.3.3 節では索引の更新を同期する場合について考慮して評価したが，その同期はどの程度の規模になるか実験を行った．実験で用いる Huffman-DHT は第 3.3.3 節と同様のものにした．図 3.10 はノード数を変化させたときの，Huffman-DHT において同じ ID が割り当てられたノードの最大数を示したものである．実験結果から，100,000 ノード規模の P2P システムにおいて最大 500 ノードの同期が必要なことがわかった．これより，Huffman-DHT における同期ノードは，十分現実的な規模で済むと言える．

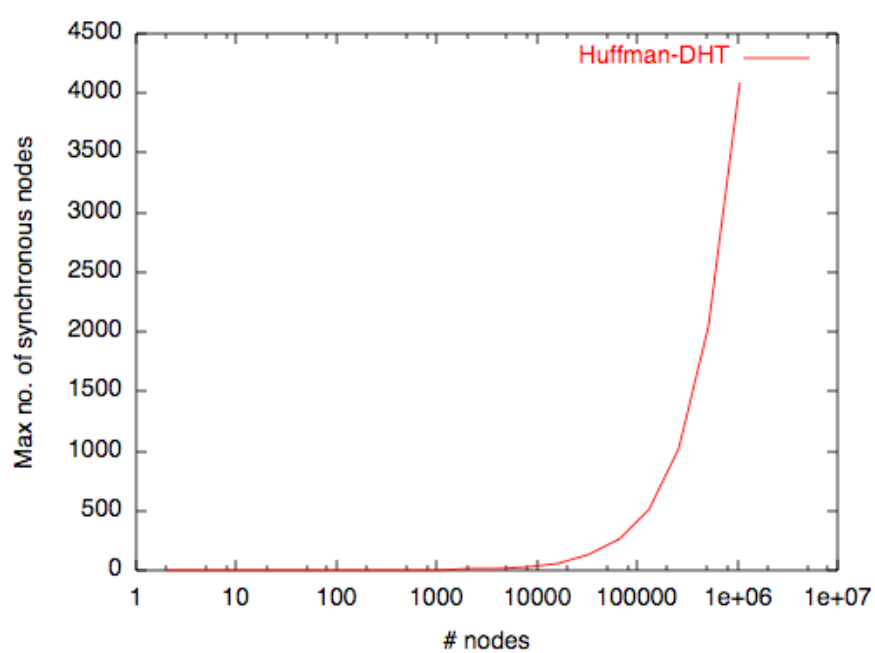


図 3.10: 索引の同期が必要なノード数の最大値

### 3.4 Huffman-DHT のまとめと今後の課題

本章では P2P 情報検索のための索引構造，Huffman-DHT を提案した．Huffman-DHT は，Huffman 符号を用いて出現確率の高い単語に対して ID 空間で広い領域を割り当てる．シミュレーションを用いた評価により，Huffman-DHT を用いることで通常の DHT を用いたときよりも索引構築の際の被アクセス数をノード間で負荷分散できることがわかった．さらに，登録頻度の高い単語の索引を保持するノードほど少ないホップ数で探せることがわかった．登録に必要なホップ数の削減については，Huffman-DHT は 100,000 ノードの規模の P2P システムにおいて最も効果を発揮することもわかった．Huffman-DHT を構築するには単語の出現確率の分布を事前に推定し，すべてのノードでそれを共有する必要がある．これについては，文書コレクションの一部の文書セットから単語の出現確率の分布が推定でき，さらに一部の頻出単語の情報だけで効率的な Huffman-DHT を構築できることを実験で示せた．

本研究における今後の課題として，検索問い合わせ処理の効率化への拡張を考えている．Huffman-DHT は文書における単語の出現確率の分布に基づいた手法となっている．この分布は検索問い合わせに出現する単語の分布とは異なると思われる．それゆえ，検索問い合わせ処理の効率化を考えたとき，この分布の不一致がどの程度影響を及ぼすか調べる必要があると思われる．

## 第 4 章

### Concordia:

### 単語の重みに基づいたデータ配置手法

## 4.1 Concordia の概要

P2P ネットワークでは、文書は分散したノードで管理される。それゆえ、P2P 情報検索ではユーザの検索問い合わせに適合する文書を分散管理された文書の中から効率良く探す機能だけでなく、ユーザの手元にそれら適合文書を効率良く収集する機能も求められる。

そこで、筆者はP2P 情報検索のための新たなデータ配置手法である Concordia を提案する。Concordia は、P2P 情報検索において P2P ネットワークからユーザの検索問い合わせに適合する文書を効率良く高速に収集することを目的とする。これを実現するため、Concordia では文書データの配置場所を検索時に索引参照のために接続する場所と関連づける。つまり、文書と関連度の高い索引を管理するノードに文書データを配置する。その関連度の計算には文書中の各単語の重みを用いる。Concordia を用いることで、問い合わせに適合する文書ほど収集が容易となり、P2P 情報検索の実行時間を削減できる。

筆者は Concordia における 2 種類のデータ配置手法を提案する。Concordia-1 は、文書における重みの大きな単語の索引ノードに文書のデータの複製を配置する。このような配置にすることで、検索問い合わせに適合する文書の収集効率を重視した配置法となっている。一方、Concordia-2 では、文書のデータの複製の代わりに Erasure 符号を用いて冗長にした分割データを配置する。Erasure 符号を用いることで、 $n$  個に分割したデータのうち任意の  $k$  個からもとの文書データを復元できる。つまり、Concordia-2 は Concordia-1 よりも、ノードの突如のネットワークからの離脱に耐えうる冗長性を確保することを目指した設計になっている。

本研究の貢献は、以下の 2 点である。

- Structured 型 P2P 情報検索の検索応答時間を削減するためにデータの収集効率に着目した。
- 単語の出現頻度の情報を P2P ネットワークにおけるデータの配置法に適用した。

本章の構成は以下の通りである。まず、第 4.2 節では Concordia のアーキテクチャについて述べる。第 4.3 節にて本手法の評価を行い、第 4.4 節でまとめと考察を行う。

## 4.2 Concordia のアーキテクチャ

Concordia の特徴は以下の通りである．

- Concordia では DHT を用いて文書ごとに単語を索引に登録することで，検索問い合わせに対する各文書との適合度を算出できる．つまり，集中型検索システムとほぼ同様の精度の検索結果を実現している．
- Concordia では文書中の単語の重みに基づいて文書データを配置するノードを決定することで，検索問い合わせに適合する文書ほど効率良く収集できる．
- Concordia-2 では文書のデータを Erasure 符号で分割することで，ノードの突如の離脱に備えた効率の良い冗長化できる．

### 4.2.1 DHT を用いた索引構造

Concordia では，ノードは DHT 上に索引を作成する．DHT はノードのアドレスとデータを共通のハッシュで管理するシステムである．ノードとデータは DHT で定められた共通のハッシュによって結びつけられており，各ノードは自分のアドレスとハッシュ値の近いデータを管理する．その結果，DHT はスケーラビリティ，耐障害性を備えた自律分散システムになっている．本研究では DHT に Kademlia[MM02] を採用した．Kademlia を用いることで，ノードは問い合わせ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．

Concordia では，この DHT を用いて文書の索引の作成と管理を行う．DHT 上の索引は各単語の索引の集合体である．各単語の索引は，単語のハッシュ値に最も近いノードが管理する．索引には 1 つの文書につき，文書名，文書における単語の重み，文書における重みの大きな上位  $n$  単語のハッシュ値，そして文書のデータへのポインタの合計 4 つの項目が登録される．

### 4.2.2 単語の重み付けと索引への登録

各ノードは索引に文書情報を登録する前に，文書における単語の重み付けの計算を行う．単語の重み付けの計算を終えた後に，各ノードは文書に含まれる単語の索引に文書名，文書における単語の重み，そして文書における重みの大きな上位  $n$  単語のハッシュ値を登録する．

この過程で用いる単語の重み付けの式は、Concordia において精度の高い検索と効率の良い収集を実現するうえで、最も重要な要素である。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。本研究では、単語の重み付けに Okapi で採用された BM25[RWJ<sup>+</sup>94] と呼ばれる確率モデルの式を基にした式を用いた [FTZ04]。文書  $d$  における単語  $t$  の重みは次のように定義する。

$$w(t, d) = \log \frac{N}{df} \cdot \frac{(k+1) \cdot tf}{k\{(1-\alpha) + \alpha \cdot \frac{dl}{avdl}\} + tf}, \quad (4.1)$$

ここで、 $w(t, d)$  は文書  $d$  における単語  $t$  の重み、 $k$  と  $\alpha$  は定数、 $N$  は文書コレクションに含まれる文書の総数、 $tf$  は  $d$  における  $t$  の出現頻度、 $df$  は全文書における  $t$  を含む文書数、 $dl$  は  $d$  の長さ、 $avdl$  は文書の長さの平均値を表す。

問い合わせ  $Q$  との適合度は以下の式を用いる。

$$R(Q, d) = \sum_{t \in Q} w(t, d). \quad (4.2)$$

P2P 環境にて式 (4.2) で示した単語の重みを計算する場合、文書の総数  $N$  と文書長の平均値  $avdl$ 、全文書におけるある単語を含む文書数  $df$ 、そして文書における単語の出現頻度  $tf$  が必要になる。P2P のような自律分散環境では、文書はネットワーク全体に分散して配置されているため、 $N$ 、 $avdl$ 、 $df$  の 3 つの値を取得することが難しい。そこで、我々は既存研究 [SMW<sup>+</sup>03] と同じように DHT を用いて構造的に単語の索引を作成し、DHT 上に配置された情報を元に  $df$  の値を得ることにした。各ノードは保持する文書すべてについて、文書に含まれる単語を担当するノードに接続し、索引に文書の情報を登録する。その結果、すべてのノードは担当する単語についての索引を管理する。ユーザは索引を参照することである単語を含む文書数、つまり  $df$  を得られる。

文書の総数は動的に変化するため、筆者らは文書の総数  $N$  と文書の長さの平均値  $avdl$  の値の算出は難しい。実験より文書の総数が検索精度に与える影響は大きくないことがわかった (第 4.3.1 節)。そこで、本研究では簡略化し、適切と思われる定数を設定することにした。

### 4.2.3 単語の重みに基づいたデータ配置

P2P IR では、検索問い合わせに適合する文書を探すだけでなく、効率良く適合する文書を収集することも重要である。既存手法では、文書を収集するたびに文書のデータを管理するノードのアドレスを ID をもとに探す必要がある。そのため、収集する文書数が多ければ多いほど多くのノードのアドレスを探す手間がかかり、検索応答に時間がかかってしまう。

そこで Concordia では、文書のデータを検索索引と関連づけることで、適合文書の収集時間の短縮を図る。筆者は、検索時に必ず接続されるノード、つまり、検索質問に含まれる単語の索引を管理するノードで、適合度の算出と文書の収集を完結できれば、文書を管理するノードを探すコストを低減できると考えた。Concordia では、文書のデータの複製を文書と関連度の高い索引を管理するノードに配置する。文書と索引の関連度の算出には文書における単語の重みを利用する。文書の検索問い合わせとの適合度は、式 (4.2) で述べたように、問い合わせに含まれる単語の文書における重みの和を用いる。それゆえ、文書における単語の重みに基づいて、データを単語の索引を管理するノードに配置することで、問い合わせとの適合度の高い文書ほど収集が容易になる。

具体的な配置の仕方は以下の通りである。

#### Concordia-1

Concordia-1 では、文書における重みの大きな上位  $n$  単語のハッシュ値に該当するノードに文書のデータの複製を配置する。Concordia-1 における索引とデータの配置法を示したものが図 4.1 である。

#### Concordia-2

Concordia-2 では文書のデータの複製の代わりに Erasure 符号を用いて冗長にした  $n$  個の分割データを配置する。Erasure 符号を用いることで、 $n$  個のうち任意の  $k$  個をデコードすることで文書のデータを得ることが出来る。Erasure 符号についての詳細は第 4.2.5 節にて説明する。Concordia-2 でもまた、文書中の単語の重みに基づいて分割データを単語の索引を管理するノードに配置する。つまり、文書において重要な単語を管理するノードほど、その文書の分割データをより多く保持することになる。筆者はある単語の索引ノードに配置する分割データの数を、文書中の

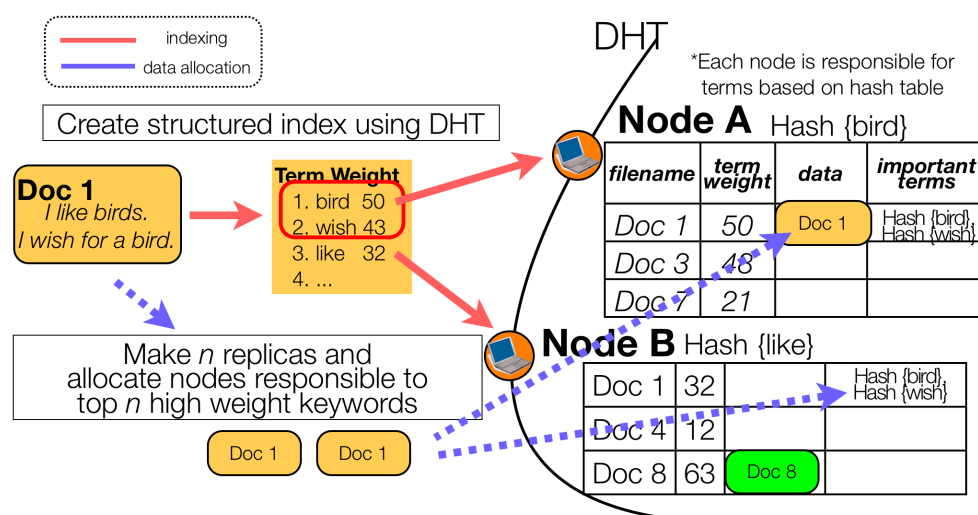


図 4.1: 索引付けとデータ配置法 (Concordia-1)

単語の重みの和のうちその単語が占める割合に比例した数に定めた。Concordia-2 は Concordia-1 に比べて、ノードの突如の離脱が頻繁に起こるような環境においても安定して文書のデータを得られる設計になっている。Concordia-2 における索引とデータの配置法を示したものが図 4.2 である。

#### 4.2.4 検索とデータの収集

ユーザが問い合わせに適合する文書を発見・収集するには、まず検索問い合わせから単語を抽出する。次に、その単語の索引を保持するノードに接続し、索引を参照する。そして、索引に登録されている文書それぞれの検索問い合わせとの適合度を求め、適合度の高い文書を収集する。その際、文書のデータは主に索引を保持するノードから収集する。もし索引を保持するノードに文書のデータが存在しない場合は、索引に登録されたその文書における重要単語を参照し、その単語の索引を担当するノードから収集を試みる。

Concordia-1 では、データ保持ノードから文書の複製を得るので、データが見つかった時点で収集の過程は終了する。一方、Concordia-2 では、データ保持ノードから文書の分割データを得るため、収集した分割データが  $k$  個に満たない場合は、索引に登録されている重みの大きな単語を参照して分割データの収集を続ける。Concordia-2 における検索と収集法を示したものが図 4.3 である。

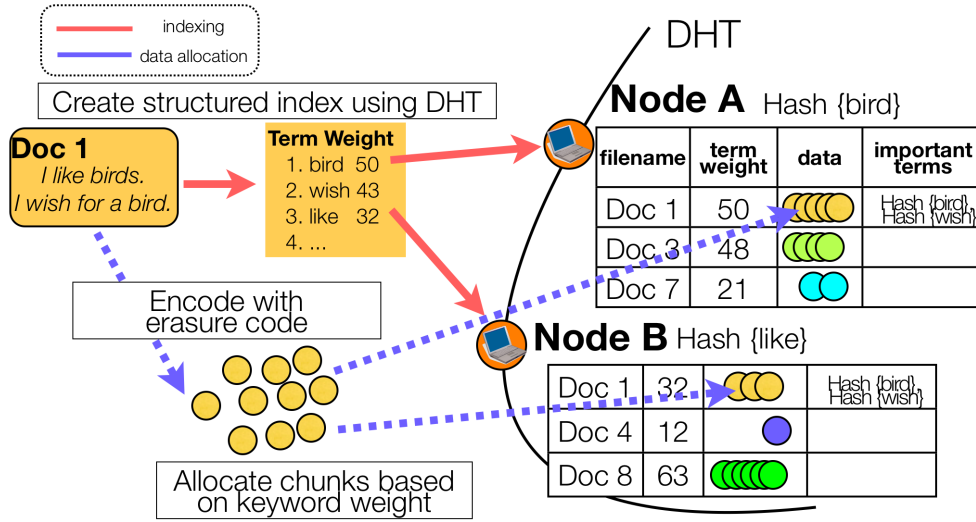


図 4.2: 索引付けとデータ配置法 (Concordia-2)

#### 4.2.5 Erasure 符号を用いた文書の分割法

Concordia-2 では、文書のデータを Erasure 符号を用いて  $n$  個に分割する。Erasure 符号を用いることで、 $n$  個の分割データのうち任意の  $k$  個から元の文書データを復元できる。つまり、仮に  $(n - k)$  個の分割データをノードの突如の離脱によって収集できなくても、ユーザはその文書を得られる。Erasure 符号には、 $(k - 1)$  次関数上の  $n$  個の座標を用いたもの [Sha79] や、 $k$  個の文字列を使った連立方程式を用いたもの [GR05] がある。いずれも文書の復元は  $k$  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 [Yam85] 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。個々の分散情報のサイズは文書のデータサイズの  $\frac{1}{L}$  となる。式で表すと以下ようになる。

$$S = S_0 \| S_1 \| S_2 \| \cdots \| S_L \quad (4.3)$$

$$y = S_0 + S_1 x + \cdots + S_L x^L + r_1 x^{L+1} + \cdots + r_{n-L} x^{k-1} \mod p, \quad (4.4)$$

$S$  は文書のデータ、 $\|$  はビット連結演算子、 $p$  は  $p > \max(S_i) (0 \leq i \leq L)$  を満たす素数を示す。分散符号化された分散情報はこの  $k - 1$  次関数上の  $(x, y)$  の座標情報  $n$  個となる。個々の分散情報のサイズ  $m$  は  $m \geq \frac{S}{L}$  となり、分散情報のサイズの総量は  $\frac{nS}{k}$  となる。データの復号に必要な計算量は、ガウスの消去法のアルゴリズムを用いた場合  $O(\log n^3)$ 、LU 分解を用いた場合  $O(\log n^2)$  となる。

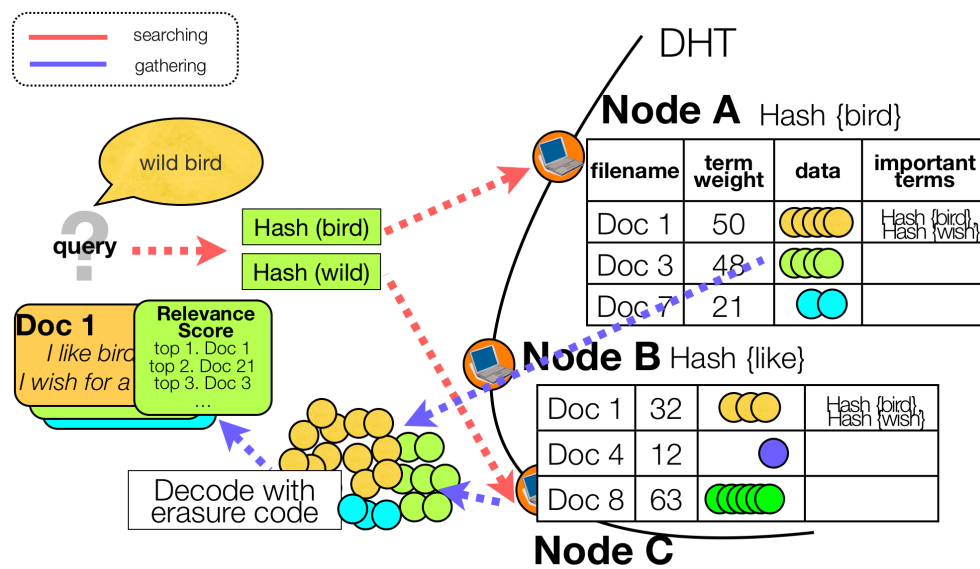


図 4.3: 検索と文書データの収集 (Concordia-2)

本研究ではこの  $(k, L, n)$  ランプ型秘密分散法を用いた． $L$  の値はデータサイズを小さくするため  $L - 1 = k$  とした．

## 4.3 Concordia の評価

Concordia は、既存研究と比べて、P2P 情報検索において問い合わせとの適合度の高い文書ほど収集を容易にすることを目指している。さらに、Concordia-2 では Erasure 符号を用いることで、突如のノードの離脱に耐えうるデータの冗長化を目指している。これらの点について評価実験を行った。

実験に使う文書コレクションとして、TREC[tre] の Text Research Collection Volume 3 (Tipster 3) を用いた。The San Jose Mercury News (1991), the Associated Press (1990), U.S. Patents (1983-1991), そして Information from the Computer Select disks (1991, 1992) copyrighted by Ziff-Davis の 4 種類からなる合計 33.6 万の文書データである。英単語のステミングには Porter stemmer[por] を用いた。ストップワードの除去を行い、ステミングをした後、式 (4.2) を用いて単語の重みの算出をした。重み付けの式のパラメータは今回は  $k$  は 100,  $\alpha$  は 0.5 を用いた。検索精度は TREC-2 ad hoc & TREC-3 routing topics のトピック 50 件を用いて求めた。トピックの例は図 4.4 である。適合率と再現率の評価には TREC Eval を使用した [tre]。

### 4.3.1 総文書数の設定が検索性能へ及ぼす影響

Concordia では文書の検索問い合わせとの適合度の計算に確率モデルの式 (4.2) を適用している。式で用いる変数のうち、文書の総数  $N$  と文書の長さの平均値  $avdl$  は処理の簡略化のため定数を設定している。定数を使用することでどの程度検索結果に影響を与えてしまうかについて、実験を行った。実験では、Tipster 3 の文書コレクションに対して TREC-2 ad hoc & TREC-3 routing topics のタスクで検索性能を調べた。

文書の総数  $N$  は式 (4.2) において、Document Frequency の値の重みに影響を与える。文書の総数を変化させたときの検索精度を示したものが図 4.5 である。図中の Baseline は正確な総文書数を用いて適合度を計算した場合の検索性能である。実験結果から、総文書数  $N$  が適合度の計算に与える影響はさほど大きくないことがわかった。大きく検索性能が低減するときは総文書数が Document Frequency の値を下回ったときであった。

文書の長さの平均値  $avdl$  が式 (4.2) に与える影響は、 $\alpha$  の設定値に依存する。この  $avdl$  と  $\alpha$  は、文書が極端に長いときに文書中の単語の重みを小さくするために用いられる。今回実験に用いた文書コレクションは文書長の差があまりないため、

```

<top>
<head> Tipster Topic Description
<num> Number: 101
<dom> Domain: Science and Technology
<title> Topic: Design of the "Star Wars" Anti-missile Defense System
<desc> Description:
Document will provide information on the proposed configuration, components, and technology of the U.S.'s "star wars" anti-missile defense system.
<smry> Summary:
Document will provide information on the proposed configuration, components, and technology of the U.S.'s "star wars" anti-missile defense system.
<narr> Narrative:
A relevant document will provide information which aids description of the design and technology to be used in the anti-missile defense system advocated by the Reagan administration, the Strategic Defense Initiative (SDI), also known as "star wars." Any reported changes to original design, or any research results which might lead to changes of constituent technologies, are also relevant documents. However, reports on political debate over the SDI, or arms control negotiations which might encompass the SDI, are NOT relevant to the science and technology focus of this topic, unless they provide specific information on design and technology.
<con> Concept(s):
1. Strategic Defense Initiative, SDI, star wars, peace shield
2. kinetic energy weapon, kinetic kill, directed energy weapon, laser, particle beam, ERIS (exoatmospheric reentry-vehicle interceptor system), phased-array radar, microwave
3. anti-satellite (ASAT) weapon, spaced-based technology, strategic defense technologies
<fac> Factor(s):
<nat> Nationality: U.S.
</nat>
<def> Definition(s):
</top>

```

図 4.4: TREC-2 ad hoc &amp; TREC-3 routing topics の一例

実験は省いた。

以上のことから，単語の重み付けの式において一部の変数を定数に設定することは検索性能にはさほど大きな影響は与えないことがわかった。

### 4.3.2 索引ノードのみから得られる文書の検索精度

Concordia では，まずはじめに問い合わせに含まれる単語のハッシュ値に対応するノードからデータを収集する．収集するデータは，Concordia-1 では文書のデータの複製であり，Concordia-2 では文書の分割データとなる．この時点で接続するノード数は，問い合わせに含まれる単語の数と一致する．この新たに文書や分散情

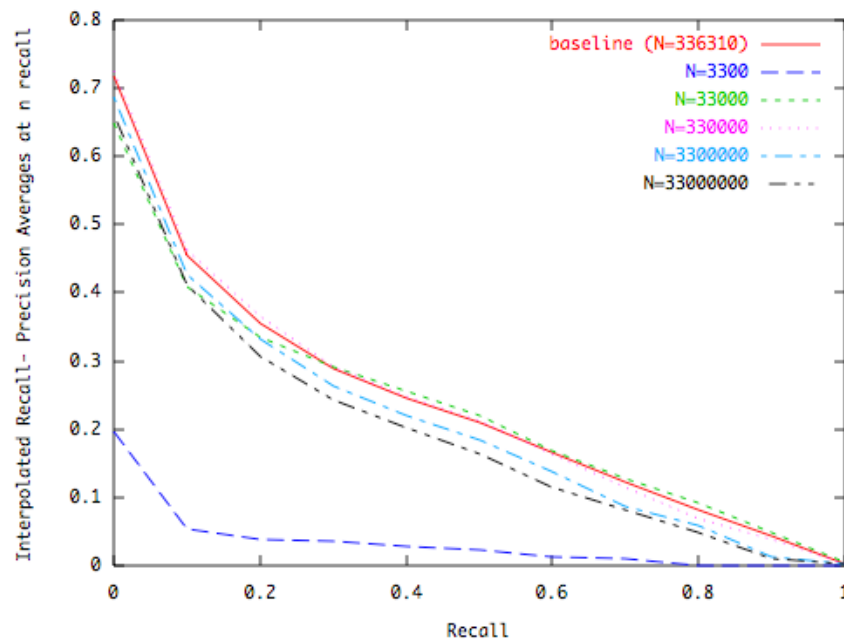


図 4.5: 総文書数の推定値を変化させたときの Interpolated Recall- Precision Average

報を収集する以前の段階で、どの程度の適合率と再現率を満たせるか、実験を行った。実験では、Tipster 3 の文書コレクションに対して TREC-2 ad hoc & TREC-3 routing topics のタスクで検索性能を調べた。Concordia-1 の結果が図 4.6 であり、Concordia-2 の結果が図 4.7 である。適合度の高い文書をすべて収集済みのときの結果を Baseline として示している。

どちらの手法も、新たに文書や分散情報を収集することのない時点でも、冗長の度合いが大きいときは高い適合率と再現率を実現できることがわかった。冗長の度合いが小さいときは、必要なデータが不十分で、新たなデータの収集が必要不可欠であることもわかった。同じだけ文書を冗長化させたときの Concordia-1 と Concordia-2 を比べると、Concordia-1 のほうが追加の収集がなくても検索性能が良いことがわかった。

### 4.3.3 適合文書の収集時に接続するノード数

Concordia を用いることで、問い合わせに適合する文書の収集が容易になるかについて実験を行った。文書の収集の容易さは、ユーザが問い合わせと適合する文

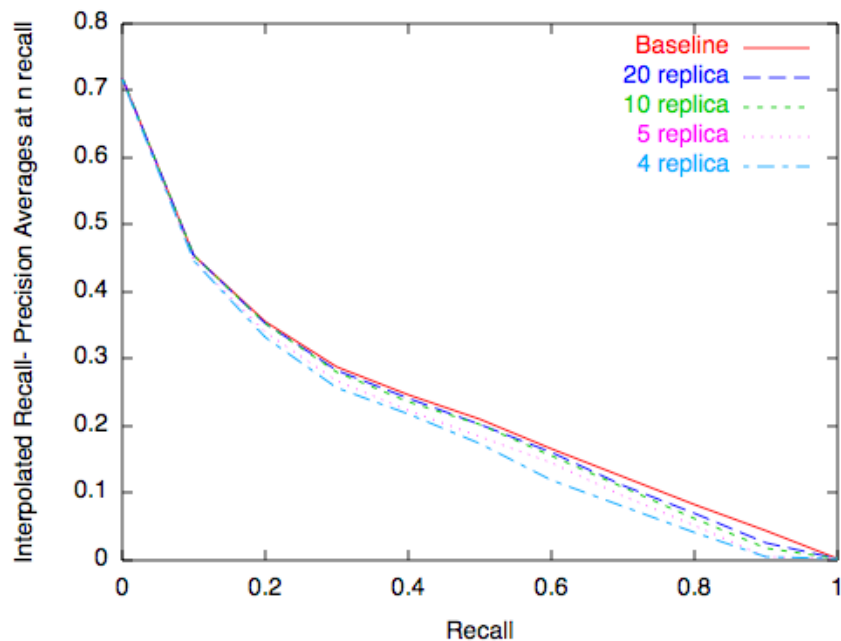


図 4.6: 索引ノードのみから得られる文書の検索精度 (Concordia-1)

書の収集課程で拡散する問い合わせ数で評価した。実験では，TREC-2 ad hoc & TREC-3 routing topics のトピック 50 件それぞれについて，適合度の順位付けを行い，適合度の上位  $n$  文書を取得するのに接続するノード数をシミュレーションした。ネットワーク内に存在するノード数は簡単のため単語数と同一の値に設定した。Concordia-1 と既存手法を比較した結果が図 4.8 であり，Concordia-2 と既存手法の比較の結果が図 4.9 である。既存手法は Baseline として示している。

実験結果より，Concordia が文書の収集の過程で接続するノード数は，問い合わせと無関係なノードに配置する手法よりも少ないことがわかった。つまり，Concordia は単語と関連するノードに文書のデータを配置することで，ユーザは問い合わせとの適合度の高い文書の収集を効率良く行える。また，冗長であればあるほど文書の収集が容易であることもわかる。同じだけ文書を冗長させたときの Concordia-1 と Concordia-2 を比べると，Concordia-1 のほうが効率良く文書を収集できることがわかった。

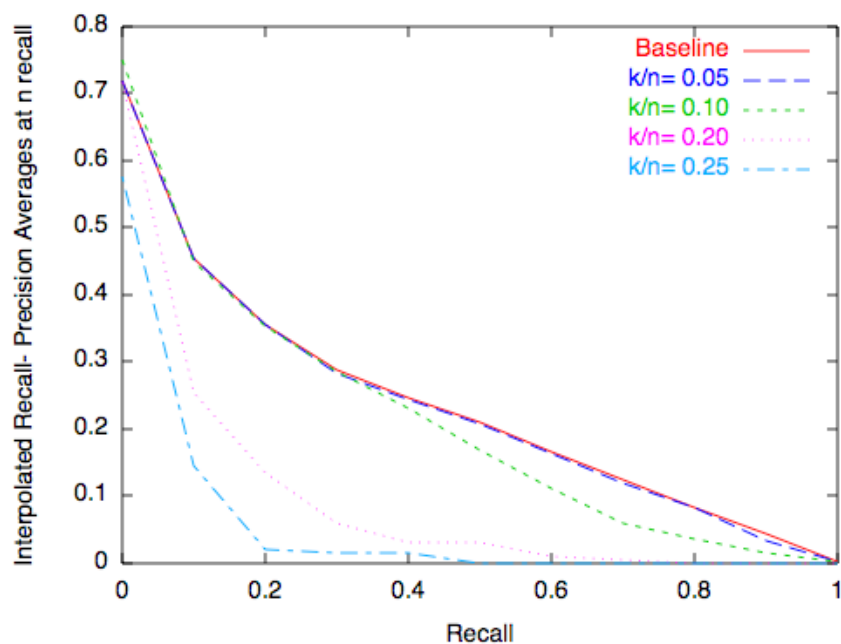


図 4.7: 索引ノードのみから得られる文書の検索精度 (Concordia-2)

#### 4.3.4 各文書における単語の重みを用いる有効性の検証

本研究の目的は、検索問い合わせとの適合度の高い文書を効率よく収集することである。これを実現するため、Concordia では各文書について単語の重みを算出し、重みの大きな単語を管理するノードに文書のデータを配置する。この手法に類似したデータ配置法として、各単語の索引で重みの大きな文書を索引ノードで保持する手法が考えられる。つまり、各索引ノードにおいて、単語の索引に登録された文書を重みでソートし、その上位  $n$  文書を予め収集する手法である。各文書における単語の重みを用いる Concordia と、各索引中の重みを用いる手法とで、検索時の収集効率を比較する実験を行った。

表 4.1 は、Tipster 3 を各手法において文書の複製を配置したときの複製数を比較したものである。Concordia-1 では各文書における重みの上位  $n$  単語の索引ノードに文書の複製を、比較手法では各索引中の重みの上位  $n$  文書の複製を配置した。単語の索引によって少ない数の文書しか登録されないものもあるため、比較手法におけるストレージコストの変化は比例になっていない。表から、各単語の索引における重みの上位 5 文書を配置するストレージコストは、Concordia において各

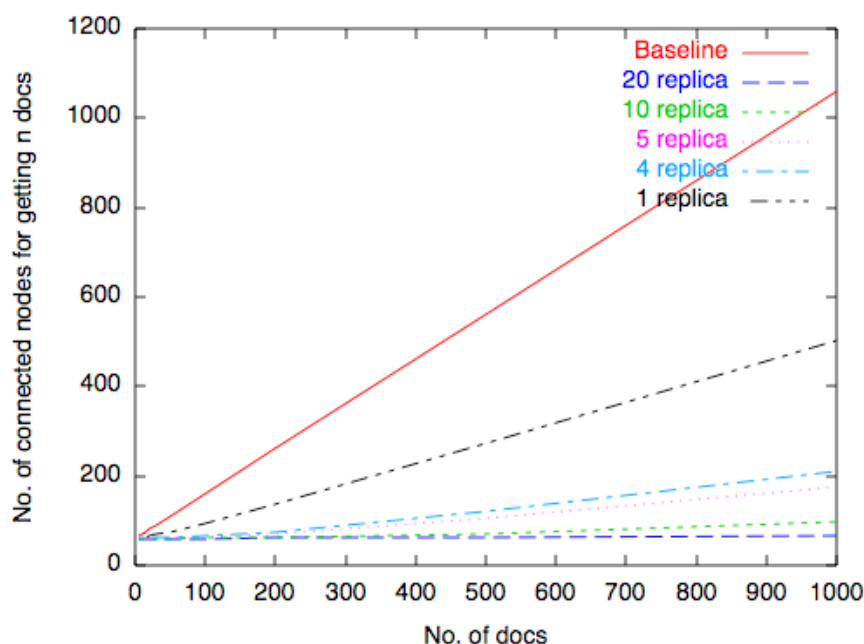


図 4.8: 適合する上位 n 文書の収集時に接続するノード数 (Concordia-1)

文書における上位 4 単語もしくは 5 単語の索引ノードに複製を配置するストレージコストとほぼ同じだとわかる。図 4.10 は、各手法で複製の配置を行った場合の単語の索引ノードに配置される複製数を比較したものである。比較手法では索引ノードに均一に複製が配置されるのに対して、Concordia では一部の索引ノードへのストレージコストが大きくなってしまふことがわかる。

図 4.11 は、第 4.3.3 節と同様の実験を用いて両手法における収集効率比較したものである。図から、Concordia のほうが冗長化の度合いが等しいとき効率の良い収集を実現できることがわかった。これは、索引中の重みに基づいた複製配置を行うと、登録数の少ない索引に複製が配置されやすくなるためだと思われる。つまり、比較手法では重みの小さな単語を管理する索引ノードに文書の複製が配置されやすくなってしまい、非効率になっていると考えられる。

以上のことから、各文書における単語の重みを用いた Concordia は、ノードに対してのストレージコストを分散する工夫が必要ではあるが、検索問い合わせとの適合度の高い文書を効率よく収集するには適した配置法であることが確認できた。

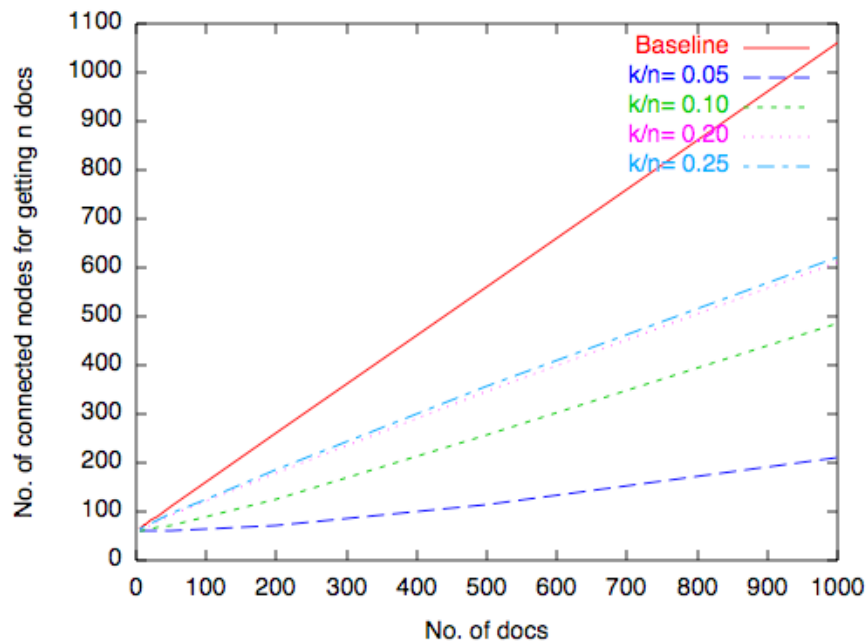


図 4.9: 適合する上位  $n$  文書の収集時に接続するノード数 (Concordia-2)

#### 4.3.5 検索応答時間 (PC クラスタを用いた実証実験)

Concordia では索引の参照と適合文書の収集を同一ノードで実行することで、検索応答時間を短縮している。筆者は分散したコンピューティング資源を用いて、Concordia-1 における検索応答時間の実測値を測り、評価を行った。実験には Tipster 3 の文書コレクションの 336,310 文書を用いた。実験環境として、InTrigger プラットフォームとして整備された 4 つの PC クラスタに含まれる合計 248 ノードで、それぞれ 5 プロセスの Concordia-1 を実行し、1,240 のノードで構成されるネットワークを構築した。実験で用いたマシンの内訳は、127 台のマシンのスペックは CPU が Pentium M 1.8 GHz で 1GB のメモリを積んだものであり、それ以外のマシンのスペックは CPU が Core 2 Duo 2.33 GHz で 4GB のメモリを積んだものであった。それぞれのマシンは Gigabit Ethernet で接続されていた。表 4.2 にマシン構成の詳細を示す。TREC-2 ad hoc & TREC-3 routing topics のトピック 50 件を検索し、適合度の上位  $n$  文書を取得するのに必要な時間を測定した。Concordia-1 で生成する複製は 5 つに設定した。単語の重みに無関係なノード 5 つに文書の複製を配置する手法を Baseline とした。図 4.12 は Concordia-1 と Baseline の検索応答時間の比

表 4.1: 配置法ごとの文書の複製数の比較

	Concordia-1	索引中の重みに基づいた複製配置
Top 1	336,310	817,897
Top 2	672,620	1,075,697
Top 3	1,008,930	1,259,765
Top 4	1,345,240	1,410,761
Top 5	1,681,550	1,542,057
Top 6	2,017,860	1,659,669
Top 7	2,354,170	1,767,310
Top 8	2,690,480	1,867,026
Top 9	3,026,790	1,960,287
Top 10	3,363,100	2,048,181

較を示したものである．

実験結果より，Concordia-1 は検索問い合わせに適合する文書を短時間で収集できることがわかった．つまり，文書中の単語の重みに基づいて文書データを配置することで，P2P ネットワークから効率良く適合文書を収集できることがわかった．実験では文書の収集に最低 180 秒必要なことがわかる．これは，索引の参照時間が大きな影響を及ぼしていると思われる．実験で使用した検索タスクは平均 62.72 単語で構成される文章であった．Concordia で検索問い合わせと文書との適合度の計算をするには，検索問い合わせに含まれる単語の数だけ索引を参照する必要がある．つまり検索問い合わせの単語数が多ければ多いほど，索引の参照時間も長くなってしまう．この問題に対しては，[PRL<sup>+</sup>07; SLZ<sup>+</sup>07] のように複数単語を管理する索引を用いることで解決できるものと思われる．

また，本実験には Concordia-2 の収集時間は測定していない．第 4.3.3 節の結果から，収集には Concordia-1 よりも多少多くの時間が必要となるが既存手法よりは効率良く収集できる．しかし，Concordia-2 では収集後に Erasure 符号で分割されたデータを復号するのに計算処理が必要となる．そのため，Concordia-1 よりも非効率な結果になると考えられる．

### 4.3.6 Erasure 符号の有効性

Concordia-2 では単純な複製ではなく Erasure 符号を用いることにより，単純な文書のデータの複製を配置するよりもノードの突如の離脱が起きる環境においても耐えうるシステムを目指している．

表 4.2: 実験で用いた InTrigger マシンの詳細

拠点	所在地	使用した ノード数	CPU	socket/ノード	core/socket	メモリ (GB)	ローカルディスク 容量 (GB)
chiba(32bit)	NII 西千葉分館	61	Pentium M 1.8GHz	1	1	1	80
chiba(64bit)	NII 西千葉分館	58	Core2Duo 2.33GHz	1	2	4	500
hongo(32bit)	東京大学 本郷キャンパス	66	Pentium M 1.8GHz	1	1	1	80
hongo(64bit)	東京大学 本郷キャンパス	13	Core2Duo 2.33GHz	1	2	4	500
okubo	早稲田大学 大久保キャンパス	14	Core2Duo Core2Duo 2.33GHz	1	2	4	500
suzuk	東工大 すすかけ台キャンパス	36	Core2Duo Core2Duo 2.33GHz	1	2	4	500

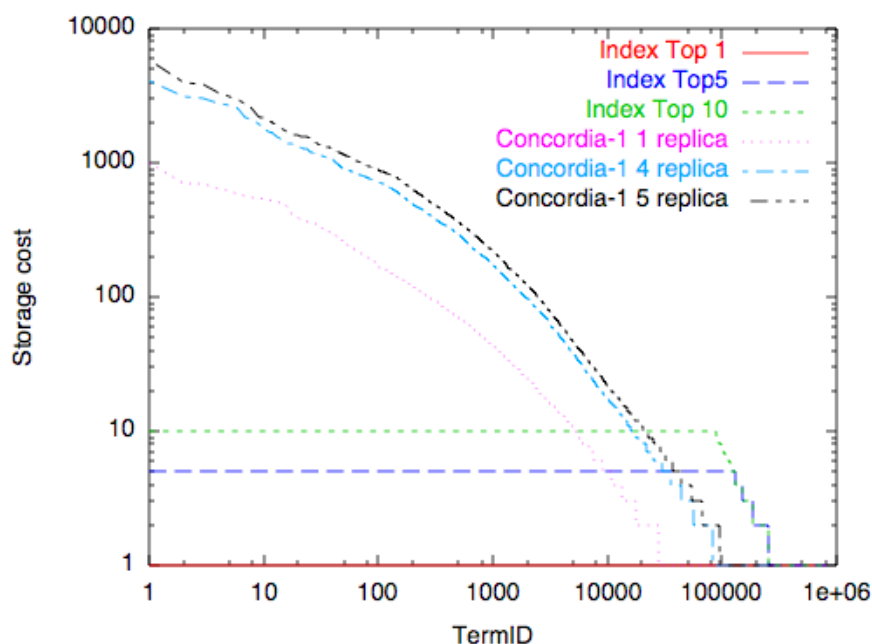
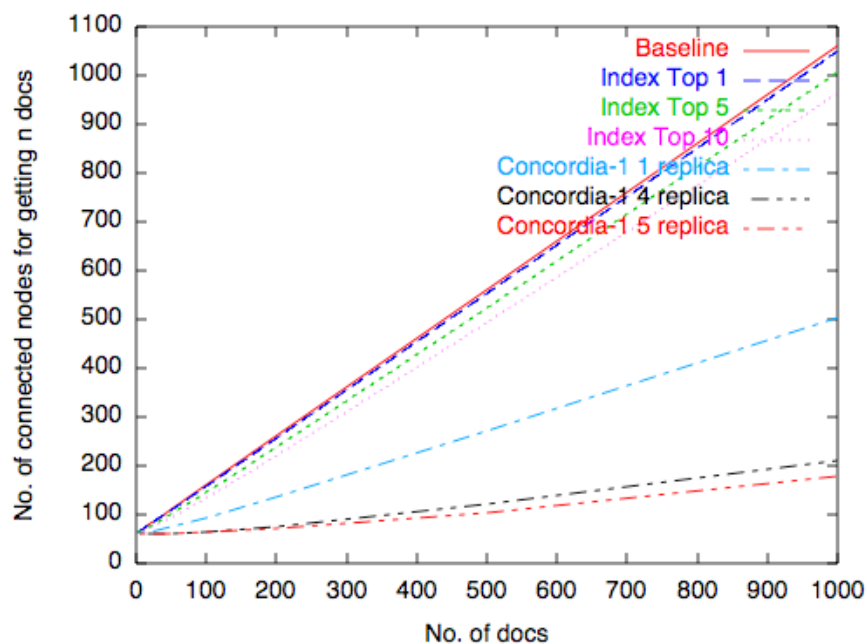


図 4.10: 各単語についてのストレージコストの比較

まず、筆者は Erasure 符号を用いたデータの分割と復号に必要な時間について実験した。実験に用いたマシンのスペックは、CPU が Core 2 Duo 2.0 GHz で 2GB のメモリを積んだものであった。Erasure 符号の処理は python で実装した。復号化のアルゴリズムはガウスの消去法を用いた。実験結果が図 4.13 と図 4.14 である。図 4.13 は分割数を一定にしてファイルサイズを変化させたときの、図 4.14 はファイルサイズを一定にして分割数を変化させたときの、分割と復号に必要な時間を示している。これらの結果から、ファイルサイズが大きいのか、もしくは分割数が多いほど、分割と復号に時間がかかることがわかった。つまり、大きなファイルを扱う P2P IR には Concordia-2 は不向きであると言える。また、Erasure 符号の分割数は P2P IR に参加するノードのマシンスペックを考慮する必要があるとわかった。

次に、Erasure 符号を用いて冗長にする有効性についての実験を行った。ノードが頻繁に離脱する状況で文書の収集がどの程度困難になるか、単純な複製を用いる Concordia-1 の冗長化と Erasure 符号を用いる Concordia-2 の冗長化とで比較したものが表 4.3 である。Concordia-2 における文書のデータの損失確率は分散情報の配分に依存する。

我々は、Concordia における文書のデータの損失確率についてシミュレーション

図 4.11: 適合する上位  $n$  文書の収集時に接続するノード数の比較

を行った．実験には，Tipster 3 に含まれる文書番号 ZF32-345-1050 を使用した．単語の重みの分布は図 4.15 である．Concordia-1 と Concordia-2 において，ノードの離脱率を変化させたときの文書の損失確率を示したものが図 4.16 である．

冗長化の度合いが同じとき，Concordia-2 のほうが Concordia-1 に比べて文書を安定して取得できることがわかった．つまり，Concordia-2 は Erasure 符号を用いることにより，単純な複製を用いる Concordia-1 よりも効率の良い冗長化が実現できていることがわかる．

表 4.3: 冗長化手法の比較

	Concordia-1	Concordia-2
Coding method	replica	Erasure code
No. of nodes holding data	$\frac{n}{k}$	$\frac{n}{k} \sim n$
File size (piece)	$F$	$\frac{F}{k}$
File size (total)	$\frac{F \cdot n}{k}$	$\frac{F \cdot n}{k}$
Document loss probability	$p^{\frac{n}{k}}$	$\sum_{i=0}^{k-1} {}_n C_i p^{n-i} \cdot (1-p)^i \sim p^{\frac{n}{k}}$

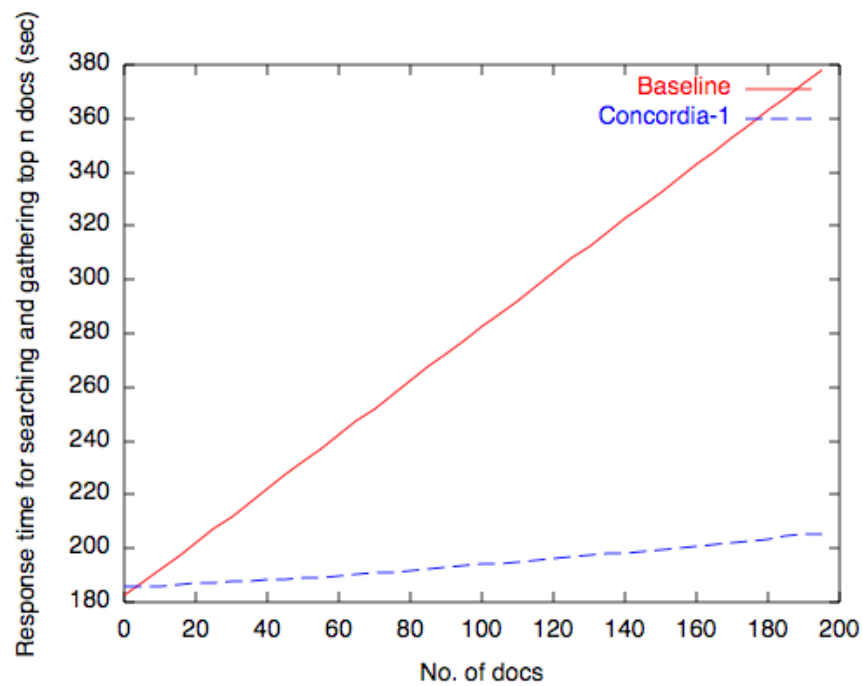


図 4.12: 適合する上位 n 文書の収集時間 (Concordia-1)

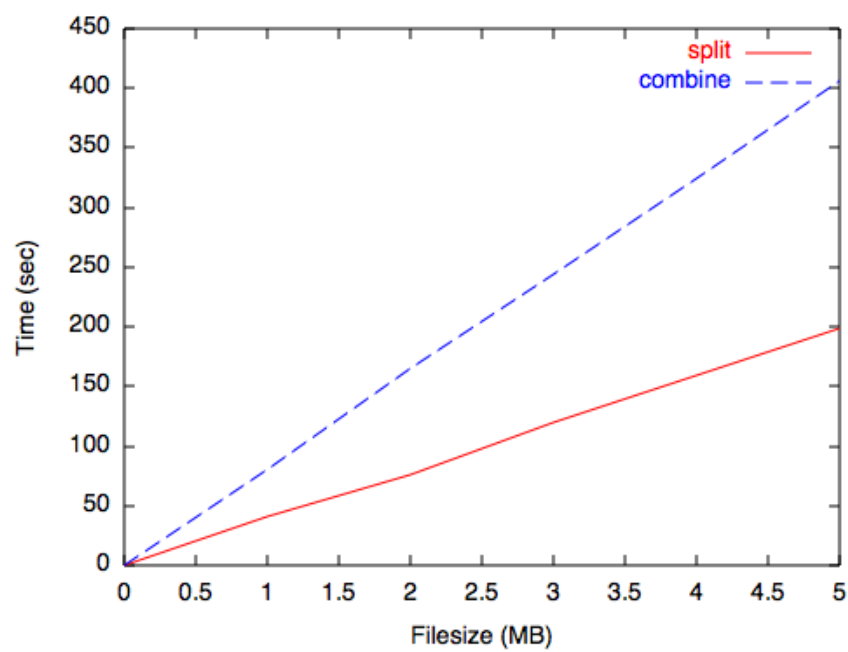


図 4.13: Erasure 符号の分割と復号に必要な計算時間 ( (20,20,100) 分割 )

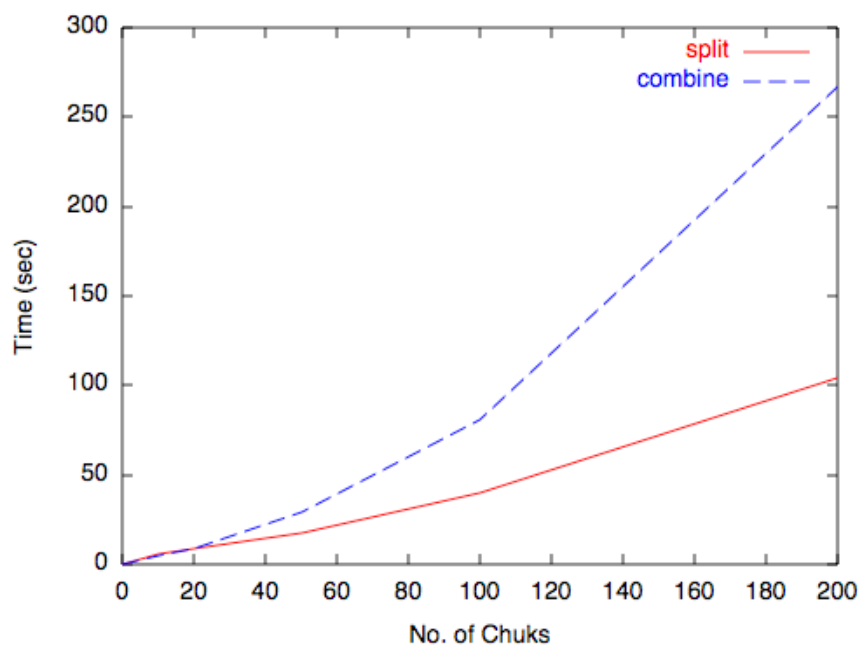


図 4.14: Erasure 符号の分割と復号に必要な計算時間 ( 1MB のファイルを (N/5,N/5,N) 分割 )

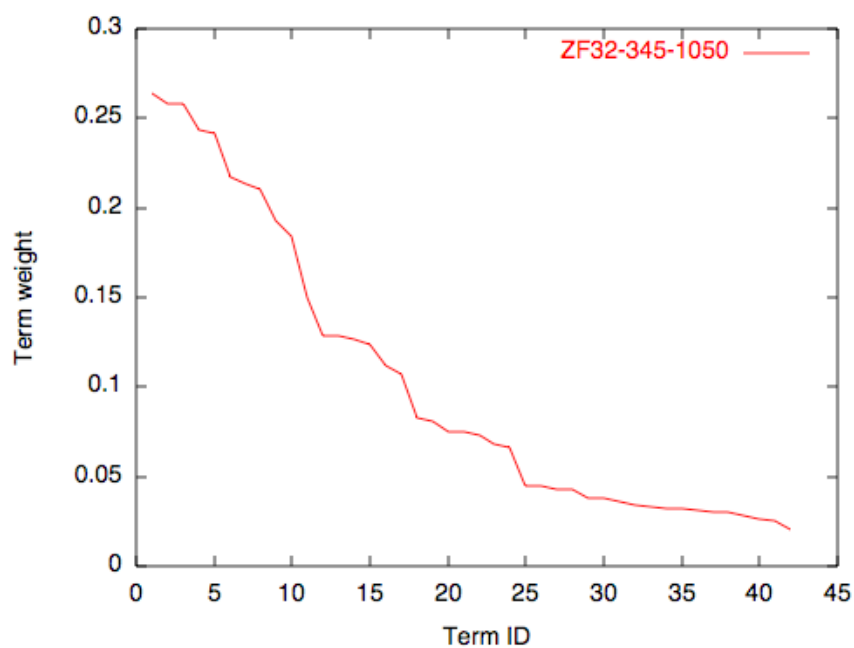


図 4.15: 単語の重みの分布 (ZF32-345-1050)

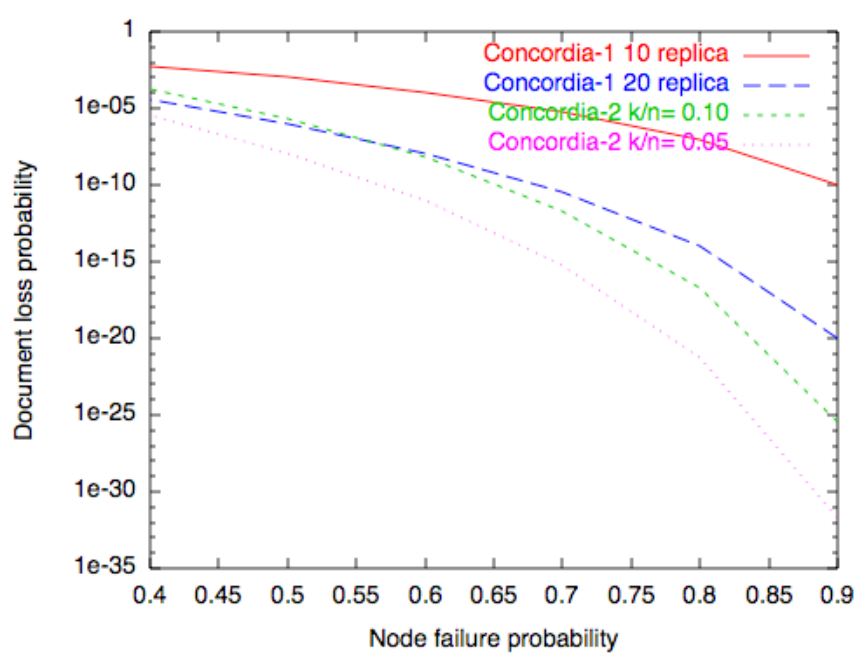


図 4.16: 文書のデータの損失確率 (ZF32-345-1050)

## 4.4 Concordia のまとめと今後の課題

本章では P2P 情報検索における索引と文書の分散配置手法, Concordia を提案した. Concordia は, 文書のデータを単語の重みに基づいて配置することで, 問い合わせとの適合度の高い文書ほど収集を容易にする手法である. Concordia-1 では, 文書における重みの大きな上位  $n$  単語のハッシュ値に該当するノードに文書のデータの複製を配置することで, 拡散する問い合わせ数を抑えた適合文書の高い収集効率を実現した. また, Concordia-2 では, 文書と関係する単語のハッシュ値に該当するノードすべてに, Erasure 符号でエンコードした分散情報を単語の重みに基づいて配置することで, ノードの突如の離脱が起こりうる状況でも安定した文書の収集を実現した.

本研究における問題点として以下の2点が挙げられる. 1つは, ストレージコストが管理する単語に依存するという点である. Concordia では複製の配置を単語の重みに基づいて行っているため, 一部のノードにデータの配置が集中してしまう. 2つ目として, ロードバランスの問題が挙げられる. 検索問い合わせに頻出する単語を管理するノードから文書のデータを収集することが多くなってしまう. これら問題点を解決するため, Huffman-DHT における技術の Concordia への転用を検討中である.

## 第 5 章

### 結論

Peer-to-Peer(P2P) ネットワークは、不特定多数の対等な立場のノードで作る分散システムのネットワークである。検索システムに P2P ネットワークを利用することで、大規模な検索基盤を容易に構築できる。その一方で、P2P 情報検索システムは、分散したノードで文書情報やデータを効率的に共有する工夫が必要となる。筆者は、索引構築のオーバーヘッドと検索実行時のコストの抑制に取り組んだ。

索引構築のオーバーヘッドを抑制するため、まず本研究にて想定する P2P 情報検索が扱う文書情報の特徴を示し、索引構築の負荷が大きくなる要因を明確にした。それを踏まえ、P2P 情報検索のための新たなノードアクセス手法である Huffman-DHT を提案した。Huffman-DHT は、Huffman 符号を用いて出現確率の高い単語に対して ID 空間で広い領域を割り当てる。この領域割り当てにより、索引登録時のホットスポットを解消し、さらに索引構築に伴う探索コストを低減する。シミュレーションを用いた実験により、索引構築負荷を分散し、少ないホップ数で索引登録できることがわかった。

また、検索実行時のコストを抑制するため、P2P 情報検索のための新たなデータ配置手法である Concordia を提案した。Concordia では文書データの配置場所を検索時に索引参照のために接続する場所と関連づけ、単語の重みに基づいて文書データを配置する。この配置法により、問い合わせに適合する文書を索引ノードから収集でき、拡散する問い合わせ数を抑えた高い収集効率を実現した。シミュレーション実験と実証実験により、問い合わせに適合する文書ほど収集が容易となり、P2P 情報検索の実行時間を削減できることがわかった。

本稿で提案した索引構造とデータ配置手法は、P2P ネットワークを用いた情報検索の効率化に有効なことがわかった。

筆者は、P2P 情報検索を実用的にすることを目標と据えている。この目標を実現するために今後取り組まねばならない課題は、以下の通りである。検索性能という観点から、情報の関連性を抽出する必要がある。集中型検索システムでは近年、Bag-of-Words の処理を発展させて高度な自然言語処理を取り入れたり、リンク情報などを用いて文書間の関連性を抽出し、検索に利用している。P2P 環境にてこれらを利用するには、分散した環境で関連性を算出する手法や、リンク情報の管理手法について検討が必要である。また、ユーザの利便性の観点から、いかなる情報も検索対象にする必要がある。本稿では文書を検索対象としていたが、画像や動画、センサ情報、ユーザコンテキストなども対象とする必要があると考えている。検索のための索引構造が異なる情報が混在していてもユーザが意識せずに検索できるような P2P 情報検索にしていきたい。今後はこれら技術課題に取り

組み，実現していきたい．

## 謝辞

本研究を行うにあたり、多くの方々にお世話になりました。

指導教員である安達淳教授には、修士研究をこのような形に至るまでの間、絶え間ない温かいご指導を頂きました。また、様々な知識に触れる機会や、充実した研究環境を与えてくださいました。お陰様で密度の濃い修士課程を過ごすことができました。深く感謝いたします。

高須淳宏教授には、修士研究における数多くのことをご指導頂き、幅広い知識をご教授頂きました。また、出張のたびにご引率いただき、お陰様で安心して発表することができました。ありがとうございます。

相澤彰子教授には、面白い分野の研究に実際に触れる機会を与えてくださいました。その経験を通じて興味分野が広がりました。ありがとうございます。

青山友紀教授、森川博之教授には、卒論でのご指導でお世話になりました。さらに、森川教授には修士1年のときに卒論の研究内容で発表する機会を与えてくださいました。初めての国際会議でとても良い経験をさせていただき、深く感謝いたします。

田浦健次朗准教授には、学部生の頃からいつも温かく接してくださり、大変お世話になりました。研究室の飲み会に参加させて頂いたり、いろいろなイベントに参加させていただき、ありがとうございます。また、修士研究の大規模分散環境での実験でもお世話になりました。近山・田浦研究室の皆様にも毎回大変お世話になりました。

浅見徹教授には、いつも温かく声をかけて頂き、大変お世話になりました。研究室に何度も遊びに行かせていただき、とても楽しかったです。ありがとうございます。浅見研究室の皆様にも大変お世話になりました。

川原圭博助教には、研究の楽しさを教えて頂き、お世話になりました。学部的时候に川原さんにご指導頂けなかったら、きっと研究を今のように楽しんで取り組めていなかったと思います。遊びに行くたびに温かく接して下さったり、公私ともに多くの助言を頂き、ありがとうございます。

猿渡俊介先輩には、学部の頃からいろいろとお世話になりました。研究につい

ても私生活の相談も、いつも親身になって接してくださいました。また、いろいろと面白いイベントに誘って頂き、とても楽しかったです。ありがとうございます。

卒論のときに所属していた青山・森川研究室の先輩や同期には、修士になってもいろいろとお世話になりました。ありがとうございます。特に、豪快で楽しい今泉英明先輩、面白い情報をいっぱい教えてくださる川西直先輩、同期の小澤政博君、ありがとうございます。

NIIで同じフロアの浅野研究室の皆様には、日々の雑談やスキー旅行など、楽しい時間をありがとうございました。OBの森本健一さん、坂巻俊明さん、そして同期の木村英雄君、西村康孝君、お世話になりました。

最後に、安達研究室の皆様には公私ともに大変お世話になりました。ありがとうございます。事務の久芳藍様には、NIIでの研究生生活を支えてくださいました。OBの正田備也助教には、データ工学の基礎知識をご教授頂きました。OGの若木裕美さんには、数え切れないほどいろいろとお世話になりました。Vu Quang Minhさん、上村明さん、相沢純也さん、辻下卓見君、辰巳正治君、広畑堅治君、大変お世話になりました。

倉沢 央

## 参考文献

- [AS07] J. Aspnes and G. Shah. Skip graphs. *ACM Trans. Algorithms*, Vol. 3, No. 4, 2007.
- [Blo70] B. H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commn. ACM*, Vol. 13, No. 7, pp. 422–426, 1970.
- [BMT<sup>+</sup>05] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. MIN-ERVA: Collaborative P2P Search. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*, 2005.
- [BMT06] M. Bender, S. Michel, and P. Triantafillou. P2P Content Search: Give the Web Back to the People. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, 2006.
- [BMTW06] M. Bender, S. Michel, P. Triantafillou, and G. Weikum. Global Document Frequency Estimation in Peer-to-Peer Web Search. In *Proceedings of 9th International Workshop on the Web and Databases (WebDB'06)*, 2006.
- [Cal00] J. Callan. Distributed Information Retrieval. *Advances in Information Retrieval, Kluwer Academic Publishers*, pp. 127–150, 2000.
- [CAPMN03] 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.
- [CRWZ05] X. Chen, S. Ren, H. Wang, and X. Zhang. SCOPE: Scalable Consistency Maintenance in Structured P2P Systems. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, 2005.
- [CSWH01] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Lecture Notes in Computer Science*, Vol. 2009, pp. 46–66, 2001.

- [FTZ04] 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.
- [gnu] *Gnutella*. <http://www.gnutella.com/>.
- [GR05] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, 2005.
- [HLW07] J. Huang, X. Li, and J. Wu. A Class-Based Search System in Unstructured P2P Networks. In *Proceedings of the 21st International Conference on Advanced Networking and Applications (AINA'07)*, 2007.
- [KM05] K. Kenthapadi and G. S. Manku. Decentralized Algorithms using both Local and Random Probes for P2P Load Balancing. In *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures (SPAA'05)*, 2005.
- [LVS<sup>+</sup>03] 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.
- [Man04] G. S. Manku. Balanced Binary Trees for ID Management and Load Balance in Distributed Hash Tables. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing (PODC'04)*, 2004.
- [MBN06] S. Michel, M. Bender, and N. Ntarmos. Discovering and Exploiting Keyword and Attribute-Value Co-occurrences to Improve P2P Routing Indices. In *Proceedings of ACM 15th Conference on Information and Knowledge Management (CIKM'06)*, 2006.
- [mec] *MeCab*. <http://mecab.sourceforge.net/>.
- [MEH05] W. Mueller, M. Eisenhardt, and A. Henrich. Scalable Summary Based Retrieval in P2P Networks. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM'05)*, 2005.
- [MM02] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Proceedings of IPTPS'02*, 2002.

- [nap] *Napster*. <http://www.napster.com/>.
- [ntc] *NTCIR*. <http://research.nii.ac.jp/ntcir/>.
- [NW03] M. Naor and U. Wieder. Novel Architectures for P2P Applications: The Continuous-Discrete Approach. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures (SPAA'03)*, 2003.
- [por] *Porter stemmer*. <http://www.tartarus.org/martin/PorterStemmer/>.
- [PRL<sup>+</sup>07] I. Podnar, M. Rajman, T. Luu, F. Klemm, and K. Aberer. Scalable Peer-to-Peer Web Retrieval with Highly Discriminative Keys. In *Proceedings of 23rd International Conference on Data Engineering (ICDE'07)*, 2007.
- [RFH<sup>+</sup>01] 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.
- [RV03] P. Reynolds and A. Vshdat. Efficient Peer-to-Peer Keyword Searching. In *Proceedings of the 2003 ACM/IFIP/USENIX International Middleware Conference (Middleware'03)*, 2003.
- [RWJ<sup>+</sup>94] 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.
- [Sha79] A. Shamir. How to Share a Secret. *Commn. ACM*, Vol. 22, No. 11, 1979.
- [SLZ<sup>+</sup>07] G. Skobeltsyn, T. Luu, I. P. Zarko, M. Rajman, and K. Aberer. Web Text Retrieval with a P2P Query-Driven Index. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '07)*, 2007.
- [SMK<sup>+</sup>01] 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.
- [SMW<sup>+</sup>03] T. Suel, C. Mathur, J. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram. ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. In *Proceedings of the 6th International Workshop on the Web and Databases (WebDB'03)*, 2003.

- [tre] *Text REtrieval Conference (TREC)*. <http://trec.nist.gov/>.
- [TXD03] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03)*, 2003.
- [Yam85] H. Yamamoto. On Secret Sharing Systems Using  $(k, l, n)$  Threshold Scheme. *IECE Trans*, Vol. J68-A, No. 9, pp. 945–952, 1985.
- [YH06] K. Yang and J. Ho. Proof: A DHT-Based Peer-to-Peer Search Engine. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI'06)*, 2006.
- [ZH05] Y. Zhu and Y. Hu. Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 4, pp. 349–361, 2005.
- [ZKJ01] B. Zhao, J. Kubiawicz, and A. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report USB//CSD-01-1141, U. C. Berkeley Technical Report, 2001.
- [ZYH05] Y. Zhu, X. Yang, and Y. Hu. Making Search Efficient on Gnutella-Like P2P Systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.

# 発表文献

## 国際会議

- [1] Hisashi Kurasawa, Hiromi Wakaki, Atsuhiko Takasu, and Jun Adachi, “Data Allocation Scheme Based on Term Weight for P2P Information Retrieval”, In *Proc. of the 9th ACM International Workshop on Web Information and Data Management (ACM WIDM 2007)*, Lisboa, Portugal, 2007. (採択率 25% (20/80))
- [2] Yoshihiro Kawahara, Hisashi Kurasawa, Hiroyuki Morikawa, and Tomonori Aoyama, “Recognizing User Context Using Mobile Handsets with Acceleration Sensor”, In *Proc. of IEEE International Conference on Portable Information Devices (IEEE Portable 2007)*, Orlando, FL U.S.A., 2007.
- [3] Hisashi Kurasawa, Yoshihiro Kawahara, Hiroyuki Morikawa, and Tomonori Aoyama, “A Dynamic User Posture Inference Scheme for Mobile Devices”, In *Proc. of 8th International Conference on Ubiquitous Computing (UbiComp2006)*, Demo Paper, Orange County, U.S.A., 2006.
- [4] Hua Si, Yoshihiro Kawahara, Hisashi Kurasawa, Hiroyuki Morikawa, and Tomonori Aoyama, “A Context-aware Collaborative Filtering Algorithm for Real World Oriented Content Delivery Service”, In *Proc. of 7th International Conference on Ubiquitous Computing (UbiComp2005)*, Metapolis and Urban Life workshop, Tokyo, Japan, 2005.

## 研究会・シンポジウム

- [5] 倉沢央, 若木裕美, 正田備也, 高須淳宏, 安達淳, “P2P 情報検索における単語の重みに基づいたデータ配置手法”, 情報処理学会 マルチメディア, 分散, 協調とモバイル (DICOMO2007) シンポジウム, 2G-4, 2007.
- [6] 倉沢央, 正田備也, 高須淳宏, 安達淳, “P2P 情報検索における索引とファイルの分散配置手法”, 情報処理学会研究報告, OS-105-21, 2007.
- [7] 倉沢央, 川原圭博, 森川博之, 青山友紀, “センサ装着場所を考慮した 3 軸加速度センサを用いた姿勢推定手法”, 情報処理学会研究報告, UBI-11-3, 2006.

## 全国大会

- [8] 倉沢央, 高須淳宏, 安達淳, “情報爆発時代における P2P 情報検索向きデータ配置手法”, 情報処理学会全国大会, 6ZK-4, 2008. (発表予定)
- [9] 倉沢央, 川原圭博, 森川博之, 青山友紀, “装着場所を考慮した 3 軸加速度センサを用いた姿勢推定手法”, 電子情報通信学会総合大会, B-15-8, 2006.
- [10] 倉沢央, 川原圭博, 森川博之, 青山友紀, “単一の無線加速度センサを用いたユーザコンテキストの推定”, 電子情報通信学会ソサイエティ大会, B-19-23, 2005.

## 表彰

- DICOMO2007 ヤングリサーチ賞, 2007
- 第 11 回 UBI 研究発表会優秀論文賞, 2006