アルゴリズム 71－1
（2000．1．17）

並列型全文検索システム構築のための手法の提案とその評価

澤田 雅人† 竹野 浩†† 藤本 典幸† 萩原 兼一†

† 大阪大学 大学院基礎工学研究科 情報数理系専攻
††NTT サイバーソリューション研究所

World Wide Web 上で文書を効率的に探しだすために全文検索システムが用いられている．ところが，近年の WWW 上で公開される文書の指数的増大により，単一計算機で構成された全文検索システムでは十分なデータ量での検索が難しくなっており，並列化された全文検索システムが構築されている．

並列型全文検索システムでは，検索に用いるデータの配置方法がスコアリングや検索速度やスループット，スケーラビリティに影響を及ぼす．そこで，本稿では，検索速度とスループットのバランスをとりつつ，スケーラビリティのある並列型全文検索システムを構築するための手法を提案し，その性能を評価する．

Proposal and Evaluation of Methods for Building Parallel Full Text Search System

Masato Sawada†, Hiroshi Takeno††, Noriyuki Fujimoto†, Kenichi Hagihara†

†Department of Informatics and Mathematical Science,
Graduate School of Engineering Science, Osaka University

††NTT CyberSolution Laboratories

Full text search systems are used to find documents on World Wide Web efficiently. However, on full text search system which is consists of single computer, it becomes difficult to search with large number of documents. So, parallel full text search systems are built.

On parallel full text search systems, a method of data arrangement has effects on search speed, throughput, and scalability. So, in this paper, we propose a method for building parallel full text search system which is well-balaced between speed and throughput and has scalability, and evaluate our method.

1 はじめに

World Wide Web(以下 WWW と呼ぶ) の出現によって，様々な文書 (以下 WWW 文書と呼ぶ) を容易に公開できるようになった．WWW 自身には検索機能や統合的な管理機能が存在しないため，ユーザは必要とする情報を得るために，何らかの手段で WWW 文書の所在 (以下 URL と呼ぶ) を入手しなければならない．このために，WWW 上から網羅的に WWW 文書を収集し，収集された文書に対して全文検索を行なう全文検索システム [1] [2] が開発されている．このシステムでは，ユーザが目的とする文書に含まれているであろうキーワードを与えることにより，ユーザが目的とする情報を持つ文書の URL を得ることができる．

ところが，近年の WWW の普及に従い，WWW 文書が指数的に増加し [3][4]，単一計算機で構成されたシステムではユーザの要求を満たすことが困難になりつつある．そこで，複数の検索サーバと検索サーバを統括する検索ゲートウェイで構成した並列型の全文検索システムの構築が行なわれている．

並列型の全文検索システムを構築する上で，検索に用いるデータの分割方法が検索速度やスループット，スケーラビリティに影響を及ぼすため，どのように分割するかが問題となってくる．

そこで，本稿では検索速度とスループットのバランスをとりつつ，スケーラビリティのある並列型全文検索システムを構築するために必要となるデータ分割の手法を提案し，その性能を評価する．

2 全文検索システム

2.1 全文検索

ある文書の集合 D に対して任意の語 w が指定されたとき，w を含む文書の集合 $A(D,w) \subseteq D$ を求めることを全文検索という．D を全文書集合，$A(D,w)$ を全文検索の結果と呼ぶこともある．

1 つの語だけでなく複数の語を指定し，それらの全文検索の結果の集合演算によって求められるように，語の指定方法を以下の検索式 q を用いて再帰的に定義する．

語 w は検索式である
q_1 および q_2 が検索式であれば
$(q_1\ and\ q_2)$, $(q_1\ or\ q_2)$, $(q_1\ not\ q_2)$
は検索式である

全文検索の結果 $A(D,w)$ を q に拡張し，以下のように再帰的に定義する．

$$A(D,q)=\begin{cases} A(D,w) & q=w \\ A(D,q_1)\cap A(D,q_2) & q=(q_1\ and\ q_2) \\ A(D,q_1)\cup A(D,q_2) & q=(q_1\ or\ q_2) \\ A(D,q_1)-A(D,q_2) & q=(q_1\ not\ q_2) \end{cases}$$

文脈より D が明確な場合，もしくは固定して考える場合は $A(D,q)$ の D を省略し $A(q)$ と書く．

2.2 WWW における全文検索システム

WWW における全文検索システムを考える場合，システムが検索対象とする文書の集合を理想的には WWW 上の全文書の集合 D_0 としたいが，現実的には収集可能な文書数の制限から D_0 のある部分集合 D_S となる．

それでも，D_S の文書数は現時点で数千万件から数億件となり，全文検索の結果も一般的に非常に多いものとなる．そのため，全文検索の結果に含まれる文書に何らかの順位をつけることが必要とされる．この順位付けを行なうため，検索結果に含まれる文書に対してスコアと呼ばれる実数値をつけるスコアリングが行なわれる．

スコアリングは唯一ではなく，現在以下のようなスコアリングが利用されている．ここで，D_0 に含まれるユニークな語全ての集合を W_0，D_0 のべき集合を 2^{D_0}，0 以上の実数の集合を R とする．

- $S_1(d,w)$
  語 w の 文書 d に対して決まるスコアで，D_0 と W_0 を定義域とし，R を値域とする関数である．
  $$S_1 : D_0 \times W_0 \rightarrow R$$
- $S_2(D_s,w)$
  語 w の文書集合 D に対して決まるスコアで，2^{D_0} と W_0 を定義域とし，R を値域とする関数である．
  $$S_2 : 2^{D_0} \times W_0 \rightarrow R$$

S_1 の例は，文書 d での語 w の出現頻度から算出する TF (Term Frequecy) 値であり，S_2 の例は，文書集合 D での語 w を含む文書の出現頻度の逆数から算出する IDF (Inverse Document Frequecy) 値である [5]．現実の全文検索システムでは，これらの値を複合的に利用して文書のスコア $S(d,q)$ を決定し，順位づけを行なっている．以降では，全文検索の結果 $A(D,q)$ は単なる文書の集合ではなく，文書とスコアの二つ組がスコアの順に並んだ系列であるとする．

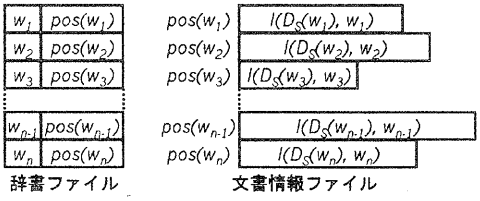

図 1: インデックスの構造

2.3 システムの記憶すべきデータ

WWW における全文検索システムでは，保有している文書集合 D_S を必ずしもそのまま記憶しておく必要はない．任意の文書 d は d の所有者の WWW サーバに記憶されており，d の所在である URL が分かれば d を取得することができるからである．以降では d の URL を $URL(d)$ とし，文書集合 D に含まれる文書の URL の集合を $URL(D)$ とする．また，文書 d の語 w に関する検索に用いる情報を $info(URL(d),w)$，w を含む文書の集合を $D(w)$，D に含まれる文書に出現するユニークな語すべての集合を $W(D)$ で表す．

語 w に関する情報の集合 $I(D(w),w)$ を以下のように定義する．

$$I(D(w),w)=\{<w,info(URL(d),w)>|d\in D(w)\}$$

全文検索システムが記憶しておくべき情報は，以下の $I(D,W(D))$ である．

$$I(D,W(D))=$$
$$\{<w,info(URL(d),w)>|w\in W(D),d\in D\}$$

実際の全文検索システムでは，実用的な検索速度を得るために $I(D,W(D))$ を構造化して記憶しており，これをインデックスと呼ぶ．

また，検索結果についても，

$$A(D,q)=\{<URL(d),S(d,q)>|d\in D\}$$

$A(D,q)$ は $<URL(d),S(d,q)>$ が $S(d,q)$ の順に並んだ系列，と定義し直せる．

2.4 インデックス

インデックスは，主に図 1の構造をした 2 つのファイルで構成される．図 1左の辞書ファイルは，語 w と 文書情報ファイル内で $I(D_S(w),w)$ を格納している場所 $pos(w)$ との対応を格納したファイルである．図 1右の文書情報ファイルには，$pos(w)$ の位置に $I(D_S(w),w)$ を連続して格納している．

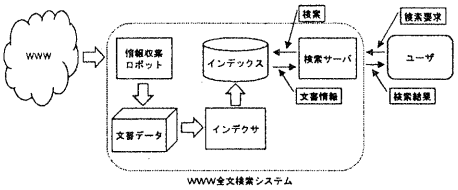

図 2: 全文検索システムの全体像

検索において語 w が与えられると，辞書ファイルを探索し，$pos(w)$ を得る．次に，$pos(w)$ を用いて文書情報ファイルから $I(D_S(w), w)$ を得る．

2.5 全文検索システムの全体像

WWW における全文検索システムでは大きく分けて，情報収集ロボットが行なう情報収集，インデクサが行なうインデックス作成，検索サーバが行なう全文検索の 3 つの処理を行なっている (図 2)．本稿ではこのうちの全文検索について詳しく述べていくが，情報収集およびインデックス作成についても簡単に触れておく．

2.5.1 情報収集

情報収集では，まず起点となる収集 URL から WWW 文書を収集する．WWW 文書には異なる WWW 文書へのリンク情報としての URL が含まれていることが多い．そこで，収集した WWW 文書から URL を抽出し，これを収集 URL 集合に加える．この処理を繰り返すことにより WWW から大量の WWW 文書を収集する．また，収集した文書から，2.5.2 で述べるインデックス作成処理を効率的に行なえるように，英単語を原形に戻す処理であるステミング処理 [5] や日本語を単語に分解する形態素解析，"a"，"the" 等のストップワード [5] を取り除く処理などを行ない単語を抽出する．こうして得られた単語と元の文書の情報等を格納した文書データをインデックス作成に利用する．

2.5.2 インデックス作成

インデックス作成 [6] では，2.5.1で収集文書を解析した結果である文書データから単語と単語が含まれていた文書情報の組を記録していく．次に，記録された単語と文書情報の組を同じ単語毎にまとめて，単語と複数の文書情報の組を作る．最後に 2.4で述べた辞書ファイルと文書情報ファイルの形に整形して出力する．

2.5.3 全文検索

ユーザからの検索要求を検索サーバが受け取ると，2.5.2で作成したインデックスから検索要求に合う文書の文書情報を得る．この文書情報から各文書のスコアを計算し，スコアの順にソートしてユーザに検索結果として返す．以降ではこの全文検索について説明を進める．

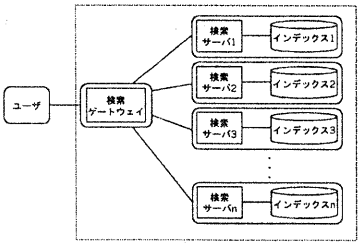

図 3: 並列型全文検索システムの構成

2.6 全文検索システムの性能指標

全文検索システムの性能を示す指標として，以下のものが挙げられる．

指標 1：検索速度
1 つの検索要求を処理するのにかかる時間で，速い方が良い．

指標 2：最大保有文書数
システムが検索対象とできる文書の最大数で，多い方が良い．

指標 3：インデックス更新コスト
インデックスに最新情報を反映させるためには，インデックスを再構築する必要がある．WWW 文書は頻繁に追加，更新，削除が行なわれる特性を持つことから，小さい方が良い．

指標 4：スループット
システムが単位時間あたりに処理できる検索要求の数で，高い方が良い．

指標 5：ディスク容量
システムがデータを記憶するのに必要なディスク容量で，少ない方が良い．

3 全文検索システムの並列化

近年の WWW 文書数の急激な増加による単一計算機での限界と，複数のワークステーションを高速なネットワークで接続した COW(Cluster Of Workstations) の利用が現実的になってきていることから，COW を利用した並列型の全文検索システムの構築が行なわれている．

COW を利用した並列型全文検索システムでは，実際の検索処理を行なう複数の検索サーバと，ユーザとの検索要求や検索結果の受渡し，検索サーバの統括を行なう検索ゲートウェイから構成される (図 3)．

ここでは，並列化を行なった場合に考えられる性能指標を挙げ，3 つの並列化手法を説明する．以降では検索サーバの台数を n 台とし，i の範囲を $1 \leq i \leq n$ とする．

3.1 並列型全文検索システムの性能指標

並列型全文検索システムの性能指標としては，2.6で述べた指標に加え，以下の指標が考えられる．

指標 6：負荷バランス
検索速度は検索サーバが扱うデータ量によって変化することから，各検索サーバの扱うインデックスのサイズにばらつきがない方が良い．また，一部の計算機に処理が集中すると，検索速度が低下したり，スループットが低下してしまう恐れがあるので，負荷のバランスが取れている方が良い．

指標 7：スケーラビリティ
計算機の追加によって，計算機の追加前に比べて扱える文書数が増加したり，検索速度が向上したり，スループットの向上が得られることで，スケーラビリティは高い方が良い．

3.2 インデックスの配置

並列化を行なう上で重要となるのが，検索対象データであるインデックスを複数の検索サーバにどのように配置するかである．インデックスの配置方法としては，インデックスを構成する要素が，文書 d 及び語 w であることから，おおまかに分けて以下の 3 つの方法が考えられる．

手法 A：同じインデックスを全計算機に配置 [1]
全文書集合 D_S から作成するインデックス $I(D_S, W(D_S))$ を全ての検索サーバに配置する．

手法 B：文書単位でインデックスを分割・配置 [7]
全文書集合 D_S を $D_{S1}, D_{S2}, ...D_{Sn}$ の互いに素な n 個の部分集合に分け，

$$I_i(D_{Si}, W(D_{Si})) = \{< w, info(URL(d), w) > | w \in W(D_{Si}), d \in D_{Si}\}$$

で定義されるインデックスを作成し，各検索サーバに配置する．

手法 C(提案手法)：単語単位でインデックスを分割・配置
$W(D_S)$ を $W_1(D_S), W_2(D_S), ...W_n(D_S)$ の互いに素な n 個の部分集合に分け，

$$I_i(D_S, W_i(D_S)) = \{< w, info(URL(d), w) > | w \in W_i(D_S), d \in D_S\}$$

で定義されるインデックスを作成し，各検索サーバに配置する．

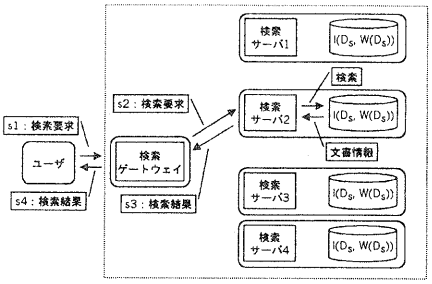

図 4: 手法 A による検索処理

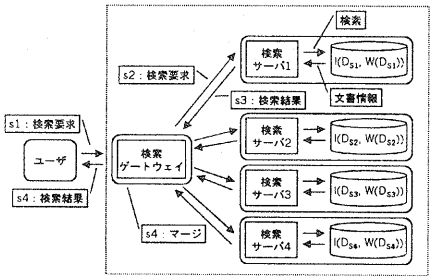

図 5: 手法 B による検索処理

3.3 検索処理

検索を行なう際，検索ゲートウェイは以下の手順で処理を行う．

s1：ユーザから検索要求の受信，解析
手法 A および手法 B では，ユーザからの検索要求 q の受信のみを行なう．

手法 C では，ユーザからの検索要求 q の受信を行ない，検索要求から語 w を抽出する．

s2：検索サーバへの検索要求の送信
手法 A では，q を送信する検索サーバを 1 台選び，そのサーバのみに q を送信する．

手法 B では，全ての検索サーバに対して q を送信する．

手法 C では，各 w について，w を含むインデックスを持つ検索サーバを求め，その検索サーバに w を検索要求として送信する．

s3：検索サーバから検索結果を受信
全ての手法において，検索要求を送信した検索サーバから検索結果を受信する．

s4：検索結果のマージ，ユーザへの送信
手法 A では，受信した検索結果が最終的な結果となっているので，そのままユーザに送信する．

手法 B では，インデックスを文書 (URL) 単位で分割していることから，それぞれの検索結果に同じ文書 (URL) に関する結果は含まれていない．そこで，受信した全ての検索結果をスコア順にソートした 1 つの検索結果にまとめるマージを行ない，その結果をユーザに送信する．

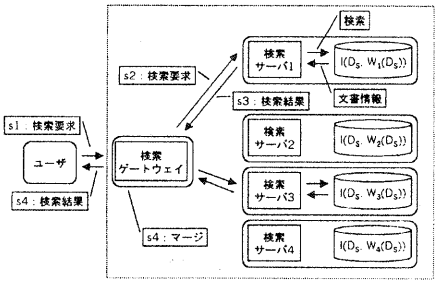

図 6: 手法 C による検索処理

手法 C では，インデックスを単語単位で分割していることから，複数の単語を含む検索要求の場合，それぞれの検索結果に同じ文書 (URL) に関する結果が含まれている可能性がある．そこで，受信した全ての検索結果を調べ，同じ文書 (URL) に関する結果を 1 つにまとめるマージを行ない，その結果をスコア順にソートしてユーザに送信する．

4 台の検索サーバを用い，検索要求に含まれる単語が 2 個の場合における各手法の検索処理の様子を図 4，図 5，図 6に図示する．

4 提案手法 (手法 C) の実装概略

手法 A 及び手法 B については，実装例 [1][7] が存在するが，本稿では手法 C における以下のようなデータ分割手法を用いた並列化手法 WDH(Word Divide by Hash) を提案する．

4.1 データ分割手法

WDH では，$W(D_S)$ に含まれる語 w のハッシュ値を用い $W(D_S)$ を $W_1(D_S), W_2(D_S), ...W_n(D_S)$ の互いに素な n 個の部分集合へ分割する．任意の w のハッシュ値は，w の文字列とハッシュ値の最大値 (以下 HMAX と呼ぶ) を与えると，0 から HMAX - 1 の間の値を返すハッシュ関数 $H(w, HMAX)$ を用いて求める．そして，

$$i = H(w, HMAX) \bmod n$$

で決まる $W_i(D_S)$ に w を割り当てる．

4.2 検索処理

3.3の検索処理のうち WDH のデータ分割手法に依存する部分について説明する．ここでは，検索要求に含まれる語数を $|q|$，検索要求に含まれる語を w_j $(1 \leq j \leq |q|)$ とする．

4.2.1 s2：検索サーバへの検索要求の送信

検索要求から抽出された語 w_j 毎に，4.1 と同じ手法で担当検索サーバを求め，検索サーバ毎に持つ検索要求リストに w_j を検索要求として加える．全ての語の担当検索サーバが求まれば，各検索要求リストの先頭の検索要求を各検索サーバに送信し，送信した検索要求をリストから取り除く．

4.2.2 s3：検索サーバから検索結果を受信

s2 で送信した検索要求に関する検索結果を受信する．送信した検索要求全ての結果を受信した後，検索要求リストに検索要求が残っている場合は s2 に戻る．

4.2.3 s4：検索結果のマージ，ユーザへの送信

3.3で述べたとおり，WDH では複数の検索サーバから得られる検索結果 $A(w_j)$ に含まれる同じ URL に関する検索結果を一つにまとめる処理を行なう．また，効率的にマージをするために，検索サーバは $A(w_j)$ をスコア順ではなく，URL の辞書式順に並べて返す．

マージ処理では，検索結果の全ての要素に対して以下の処理を行なう．検索結果が URL の辞書式順に並んでいることから，2 本のリストの先頭の要素同士を比較し，同じ URL に関する検索結果であればマージを行ない，スコアが 0 でなければマージ結果のリストに付け加える．異なる URL に関する検索結果であれば，辞書式順で先にくる URL の要素と同じ URL でスコアが 0 の仮要素とでマージを行ない，スコアが 0 でなければマージ結果のリストに付け加える．マージを行なった要素は元のリストから取り除く．

検索結果を $< URL(d), S(d,q) >$ が $URL(d)$ の辞書式順に並んだリスト，検索要求を $q = q_l \; op \; q_r$ (op は and, or, not のどれか) とする．q_l および q_l それぞれの検索結果のリストの先頭の要素を $< URL(d_l), S(d_l, q_l) >$，$< URL(d_r), S(d_r, q_r) >$ とする．マージ処理は以下の疑似プログラムに示す方法で行なう．

```
if (URL(d_l) == URL(d_r)) then {
  URL(d) = URL(d_l);
  S_L = S(d_l, q_l);
  S_R = S(d_r, q_r);
} else if (URL(d_l) ≤ URL(d_r)) then {
  URL(d) = URL(d_l);
  S_L = S(d_l, q_l);
  S_R = 0;
} else {
  URL(d) = URL(d_r);
  S_L = 0;
  S_R = S(d_r, q_r);
};
if (op == and) then {
  S(d,q) = min(S_L, S_R);
} else if (op == or) then {
  S(d,q) = S_L + S_R;
} else {
  if (S_R == 0) then {
    S(d,q) = S_L;
  } else {
    S(d,q) = 0;
  };
};
```

この結果得られた $URL(d)$ とスコア $S(d,q)$ の組

において $S(d,q)$ が 0 でなければ，マージ結果のリストに追加する．これを全ての検索結果が 1 つのリストになるまで繰り返し，最終的に得られたリストをスコア順にソートして，検索結果 $A(q)$ とする．この検索結果をユーザに送信する．

5 評価実験

2.6および 3.1における各性能指標のうち，指標 1，指標 5，指標 6，指標 7 について，WDH の評価を行なうための実験を行ない考察した．また，指標 2，指標 3，指標 4 については，実験結果を用いて考察を行なった．

5.1 実験環境

実験環境は，

CPU	PentiumII 350MHz
メモリ	512MB
ハードディスク	UW-SCSI 9.1GB
OS	Solaris 2.6 for Intel

で構成した計算機 8 台

CPU	PentiumII 450MHz
メモリ	384MB
ハードディスク	U2W-SCSI 18.2GB
OS	Linux 2.0.38

で構成した計算機 1 台をすべて 100Mbps のイーサネットスイッチングハブで接続した COW を利用した．

5.2 実験内容

性能指標毎に以下の実験を行なった．用いた文書集合は WWW より無作為に収集した約 50 万件の文書である．

指標 1：検索サーバを 8 台利用し，複数の検索要求を用いて検索を行ない，検索結果の件数と検索処理にかかる時間の関係を測定する．

指標 5：各手法において検索サーバが 2 台の場合から 8 台の場合までで，生成されたインデックスのサイズおよび，そのばらつきを調べる．

指標 6：手法 B においては，インデックスサイズのばらつきで，WDH においては，実際の検索システムでの検索要求に含まれていた単語 (重複を含む約 50 万語) を，提案手法のハッシュ関数でハッシュし，そのハッシュ値のばらつきを調べることで負荷の偏りを調べる．

指標 7：検索サーバを 2 台から 8 台まで変化させたときに，指標 1 の実験に用いた検索要求に対して検索処理にかかる時間の変化を測定する．

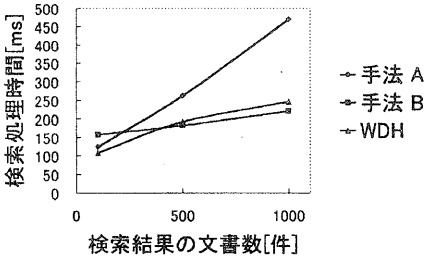

図 7: 検索速度

5.3 実験結果及び考察

指標 1：図 7は検索結果の件数を変化させた場合の，各手法での検索処理にかかる時間のグラフである．手法 A では，件数の増加に伴い検索処理にかかる時間がほぼ線形に増加している．これは件数の増加に伴い結果の通信量が線形に増加するからである．手法 B では，並列度が n であるが，件数の少ないうちは全ての検索サーバと通信を行なうことによる通信コストが並列化による検索速度の向上を上回ってしまう．件数が増加すれば検索処理全体に占める検索サーバでの処理の割合が高くなるため，並列化の効果が得られるようになる．WDH では，最大で $\overline{|q|}$ 台の検索サーバと通信すればよい．実際の検索システムに送られた検索要求について含まれる語の数を調べたところ，平均で約 3 個であった．このことから，通信コストが低く抑えられ，件数が少ない場合も並列化の効果が得られる．しかし，得られる並列度が最大でも $\overline{|q|}$ までであり，実際は $\overline{|q|}$ は高々 3 であることから，件数が増加すると検索処理全体に占める検索サーバでの処理の割合が高くなるので，手法 B よりも処理時間が長くなってしまう．

指標 5：

作成したインデックスのサイズの合計，平均，標準偏差を表 1に示す．

手法 A では 1 つのインデックスの複製を各検索サーバに置くため，検索サーバの台数の増加に伴い必要なディスク容量が線形に増加する．手法 B 及び WDH では，インデックスを構成する要素を複数のインデックスに分割することから，検索サーバの台数の増加による必要なディスク容量の増加は比較的少ない．この増加は，インデックスの分割によって複数のインデックスに重複した情報が存在するために発生する．したがって，分割数が増えると重複する情報の割合も高くなるため必要とするディスク容量も増加する．

表 1: インデックスサイズの合計/平均/標準偏差

	2 台	4 台	8 台
手法 A	3006.8MB 1503.4MB 0	6013.6MB 1503.4MB 0	12027.2MB 1503.4MB 0
手法 B	1534.0MB 767.0MB 24.7	1573.8MB 393.5MB 23.5	2339.1MB 292.4MB 57.5
WDH	1580.2MB 790.1MB 4.2	1711.6MB 427.9MB 9.5	2208.9MB 276.1MB 37.4

上段:合計，中段:平均，下段:標準偏差

表 2: WDH(8 台) の各ファイルサイズ (MB)

	辞書ファイル	文書情報ファイル
検索サーバ 1	7.2	175.0
検索サーバ 2	6.7	170.0
検索サーバ 3	6.7	158.1
検索サーバ 4	6.7	228.8
検索サーバ 5	6.7	170.4
検索サーバ 6	6.8	153.8
検索サーバ 7	6.7	155.4
検索サーバ 8	6.6	165.7

インデックスサイズのばらつきについては，手法 A では，手法から明らかにばらつきは発生しない．手法 B では，文書数がほぼ均等になるように分割を行なっているため，各文書のサイズの違いから若干ばらつきが生じている．WDH では単語単位で分割をしており，ハッシュを用いることでほぼ均等に割り振ることができているため，ばらつきが小さくなっている．しかし，8 台に分割した場合のばらつきが少し大きくなっているのは，表2より，単語数に比例する辞書ファイルがほぼ均等になっていても，単語を含む文書数のばらつきが大きい場合には，文書情報ファイルに偏りが生じるためである．

指標 6:手法 A では，全ての検索サーバが同じ構成となるため，ラウンドロビンで担当検索サーバを決定すれば負荷の偏りは生じない．手法 B では，表1からインデックスサイズの大きなばらつきがないことより，負荷に偏りが生じることはないといえる．WDH におけるハッシュによる検索要求に含まれる単語の散らばり具合について表 3に示す．各検索サーバに送られる単語の数には大きなばらつきがないことから，負荷に偏りが生じることはないといえる．

表 3: 検索要求中の単語のばらつき

	2 台	4 台	8 台
検索サーバ 1	255379	128866	65504
検索サーバ 2	254770	127087	63209
検索サーバ 3		126513	62722
検索サーバ 4		127683	63485
検索サーバ 5			63362
検索サーバ 6			63878
検索サーバ 7			63791
検索サーバ 8			64198

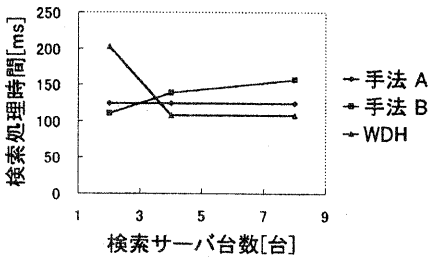

図 8: 検索サーバ台数による検索処理時間の変化

指標 7：図 8は検索サーバの台数を 2，4，8 台と変化させ，同じ検索要求を処理したときの検索時間を表している．手法 A では，検索サーバの台数が変化しても利用する検索サーバの台数が 1 台であるため，検索速度の変化はない．手法 B では，検索サーバの台数の増加によって，通信コストが増加するため並列化の効果を相殺してしまう．これは，検索件数が少ない場合に顕著であるが，台数がさらに増加すると並列化の効果が得られなくなることも考えられる．WDH では，$\overline{|q|}$ によって通信する検索サーバの台数および並列度が決まるため，検索サーバが少ない場合に同じ検索サーバに複数の語が割り当たる確率が高くなるため，検索速度の低下起こる (図 8の 2 台の場合)．しかし，検索サーバの台数が増加すると，同じ検索サーバに複数の語が割り当たる確率が低くなり，通信する検索サーバの台数および並列度は変化しなくなるので，検索サーバの増加による検索速度の変化がほとんどなくなる．

以下の指標については，実験結果および手法の特徴を用いて考察する．

指標 2：手法 A での最大保有文書数を N とする．表 1より，手法 B では，2 台で約 1.96N，4 台で約 3.82N，8 台で約 5.14N の文書を保有できる．WDH では，2 台で約 1.90N，4 台

表 4: 性能指標における各手法の比較

	手法 A	手法 B	手法 C
指標 1	×	○	△
指標 2	×	○	○
指標 3	×	○	×
指標 4	○	×	○
指標 5	×	○	○
指標 6	○	○	○
指標 7	×	△	○

で約 3.51N，8 台で約 5.45N の文書を扱えることになる．手法 B と WDH を比較すると，台数が少ないうちは手法 B の方が多くの文書を保有できるが，台数が多くなるにつれて WDH の方が多くの文書を保有できるようになる．これは，手法 B は分割数が増えるにしたがって情報が重複する確率が高くなってしまうが，WDH は台数が少ないうちは重複する確率がやや高いが，分割数の増加による確率上昇が少ないからだと考えられる．

指標 3：手法 A では，ある 1 つの d の更新を反映するだけでもインデックスを最初から作り直す必要がある．手法 B では，ある 1 つの d の更新を反映するだけならば，d を含む D_{Si} から作られたインデックスを再構築するだけでよい．WDH では，更新された d に含まれている単語を含むインデックスの更新が必要で，最悪の場合全てのインデックスを再構築する必要がある．

指標 4：検索サーバの台数を n 台，1 つの検索処理に必要な検索サーバの台数を p 台，検索要求当たりの処理時間を t [s] とすると，スループット TP は以下の式で求めることができる．

$$TP = (n \div p) \times (1 \div t)$$

図 7で検索結果の件数が 1000 件の場合で 1 秒当たりに処理できる検索要求数を計算すると，

手法 A では，$(8 \div 1) \times (1 \div 0.47) = 17.0$
手法 B では，$(8 \div 8) \times (1 \div 0.22) = 4.5$
WDH では，$(8 \div 3) \times (1 \div 0.25)) = 10.7$
となる．

5.4 総合評価

手法 A はスループットは優れているが，最大保有文書数や検索速度の点で問題がある．手法 B は検索速度やインデックスの再構築のコストの点で優れているが，スループットの低さや検索速度に関するスケーラビリティの点で問題がある．WDH はスループットや検索速度に関するスケーラビリティは優れているが，インデックスの再構築のコストがかかる点や，検索速度が手法 B に比べるとやや劣る点に問題がある．以上をまとめると，表 4になる．WWW における全文検索システムでは扱う文書数が膨大であることから，手法 A は適していないと考えられる．手法 B と WDH では，検索速度に重点を置くなら手法 B が，スループットやスケーラビリティに重点を置くなら WDH が適していると考えられる．

6 まとめと今後の課題

並列型全文検索システムを構築する上で重要となるインデックスの分割について 3 つの手法を示した．そのうち，単語単位での分割手法である手法 C は我々が新たに提案したものである．また，これら 3 つの手法についての性能評価を行なった．

今後の課題として，今回評価実験を行なわなかった性能指標に関する検証実験および，手法 C における欠点の解消が挙げられる．手法 C における性能指標 3 に関する欠点は，並列化手法 B を併用することで解消できると考えられ，現在実装および実験を進めている．

7 謝辞

本研究は一部平成１１～１２年度文部省科学研究費補助金・基盤研究 (C) (11680357) およびＰＤＣ (並列・分散処理研究推進機構) の補助による．

参考文献

[1] AltaVista : http://www.altavista.com/

[2] goo : http://www.goo.ne.jp/

[3] Steve Lawrence and C. Lee Giles : Searching the World Wide Web, *Science*, No.280, pp. 98-100, 1998

[4] 内田 斉, 宮沢 浩, 大岩 寛 : 我が国の WWW コンテンツ量の推計, http://www.a-brain.com/WebCount/index.html

[5] Frakes, William B., Baeza-Yates, R.: "Information Retrieval : data structure and algorithms" Prentice Hall, 1992

[6] Salton, G. and McGill, M. J.: Introduction to Modern Information Retrieval, McGraw-Hill, chapter2, 1983

[7] Cahoon, B., McKinley, Kathryn S.: "Performance Evaluation of a Distributed Architecture for Information Retrieval", 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 110-118, 1996