D-031

ログ構造の分割インデックスシステム
Partitioned Indexes System with Log-structure

菅 豊† 佐藤 隆士‡
Yutaka Kan Takashi Sato

1. はじめに

今日では，新聞や特許，WEB，XML などの大規模な文書集合から，目的とする文書にアクセスするために検索の必要性が増加している．このような文書を効率的に検索しアクセスするためには，文書に対してインデックスを作成するのが一般的である．しかし，文書集合の巨大化に伴い，インデックス作成や文書更新におけるインデックスの保守が以前にも増して負担になっている．

これらの文書集合を更新する際には，新しい文書の追加を行うことが主で修正や削除は比較的少なく，古い文書の修正はほとんどないなどの特徴的パターンが認められる．従って，このような特性に着目したインデックスの構成法を考えなければならない．

一方，インデックスの処理装置であるコンピュータの性能は向上しているが，単一処理装置の飛躍的な性能向上は期待できない状況である．当面の性能向上は，CPU のマルチコア化，コンピュータのクラスタ化などであり，分散並列処理に頼らざるを得ない．

このような状況から，本稿では以前より，大規模文書集合の特性と最近のコンピュータの構成に適合した，ログ構造を有する分割インデックス[1]の提案をしてきた．

ハードディスクのアクセス特性に合わせて，ファイルの書き込みを特定部分に集中することにより，性能向上を目指したログ構造ファイルシステム[2]になぞらえ，構築するインデックスシステムをログ構造の分割インデックスシステムと呼ぶことにする．

2. インデックスの構成

文書集合をコーパス C とし，その分割を $C_i(i=1,2,...,m)$ とする．C_i の転置ファイル $T_i(=C_i^I)$の集合 $T=\{T_1,T_2,...,T_m\}$ が提案の分割インデックスである．ここで，コーパス C は要素となる文書の作成時間順（到着順）にグループ化され C_i に分割されるものとする．分割インデックスはこのグループごとの小インデックスである．図 1 に構成を示す．図中 S は，追加文書用の分割インデックスである．

通常のインデックスを転置ファイル（Inverted File）とも言うが，分割インデックスは部分転置ファイル（Partial Inverted File）に相当する．

転置の方法は応用によって異なる．例えば全文検索の場合は，文書を構成する語を見出しに文書 ID を求める．リレーショナルデータベースでは属性値を見出しにタプル ID を求めるなどである．

文書の作成順に一定数のグループごとに小インデックスを作成する．つまり，グループごとに独立にインデックス作成が可能なため，容易に分散並列で処理できる．

†大阪教育大学大学院 教育学研究科 総合基礎科学専攻 数理情報コース

‡大阪教育大学 情報処理センター

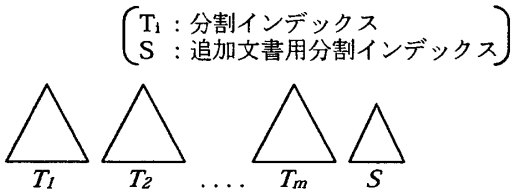

図 1 ログ構造分割インデックスの構成

3. 検索と更新処理

3.1 検索処理

分割インデックスでは，分割しない単一のインデックスに比べ，それぞれのサイズが小さいため高速な探索が期待できる．各分割インデックスを担当する複数の処理装置に，クエリをマルチキャストして探索させ，目的の文書 ID を取得する．複数の分割インデックスに対する探索が必要となるが，分散処理で同時に探索可能である．

この場合，分割インデックスの探索結果をマージする時点で特定の処理装置に負荷が集中することになるが，通常のクエリでは探索結果の集合がコーパスに対して格段に小さいため，問題になることはほとんどない．また探索後も引き続き分散して処理する場合，検索段階でのマージは不要になる．

3.2 更新処理

更新操作は，追加，修正，削除からなるが，本稿では議論を単純にするため，修正は削除と追加の組み合わせで実現することとし，扱わないことにする．

3.2.1 追加

追加は文書の最終グループ用のインデックスに対して行う．この処理は分散しない場合に比べ，インデックスサイズが小さいため高速である．

追加により最終グループが他のグループと同じ程度の大きさになった場合は，新グループを作成して索引を追加する．その結果，今までの最終グループは，最後から 2 番目のグループになる．

また，最終文書グループ以外は別の分割インデックスで索引付けられているため，これらのインデックスを使用した検索は，文書追加で発生するインデックスの更新処理を待つことなく実行可能である．即ち，検索結果に必ずしも最終の文書グループを必要としない場合は，更新と検索を並行に処理できる．

データの到着順ではなく，データベースの値で分割した場合は，このような検索と更新の独立性が保障されないため，並行処理は困難である．

3.2.2 削除

文書の削除時には，削除表に文書 ID を記入するのみで，インデックスに反映しない方式とする．その際，各分割

インデックスを担当する処理装置ごとに削除表を用意し，メモリ上に置く．追加することが主で，修正や削除は比較的少ない場合を想定しているため，削除表による使用領域の増加は緩やかである．

検索時には，インデックスを用いた検索をした後，削除表に含まれる文書 ID を除外する必要がある．この処理は検索時のオーバヘッドになるが，削除表をメモリ上に置くため，他の処理に比べて無視できる程度となる．

また，長期間にわたる削除により削除表が負担となる場合，計算リソースが空いているタイミングで，分割インデックスを更新して削除表を空にする．この処理は稀なため，日常の検索，追加の処理と比較すると，無視できる程度の割合である．

4. 実験結果

国立情報学研究所における情報検索システム評価用テストコレクション（NTCIR）から NTCIR-6 言語横断検索（CLIR）タスク[3]に用いられた新聞 2 紙の 2 年間（2000-2001），計 4 年分 858,400 件の記事の本文（約 1GB）をコーパスとしてインデックス作成実験を行った．索引語には可変長グラム[4]を使用しており，その総数は約 4 億（398,933,412）である．使用コンピュータ（CPU：4 コア，HDD：SATA×4 台）と OS は以下の通りである．

CPU : Intel® Core™2 Quad CPU Q6600 @2.40GHz
Memory : 3G
Disk : Seagate ST3250410AS-BOX×4
OS : FreeBSD 7.0-RELEASE

なお，インデックス格納には Berkeley DB4.3.26[5]をアクセス法 B-tree で利用している．

本稿では比較のために，文書集合を 2 分割，4 分割したものを各ディスク上で分散並列処理した，ログ構造の分割インデックスと分割しない単一インデックスの 3 種類における，作成と検索の性能を測定した．

4.1 作成

単一インデックスと分割インデックスの作成時間とサイズを表 1 に示す．分割インデックスの行には，2 分割と 4 分割の分割インデックスを分散並列により作成した際の最長時間と各サイズの総和を記載している．ここで，分割インデックスの作成時間を平均すると 2 分割では 61.3 分，4 分割では 33.2 分となったが，本稿では並列に分割インデックスを作成したため，作成時間を平均ではなく最長時間としている．この平均と最長における時間差は，メモリ領域やディスク書き込み処理などの制限により，処理待ちが生じた結果である．

分割インデックスにすることにより，インデックスの作成時間を大幅に短縮することができたが，分割インデックスの総和サイズは，単一インデックスよりも大きくなった．これは，文書集合を単純に 2 分割，4 分割したため，各分割インデックスにおいて，重複するデータ項目が存在しているためと考えられる．

表 1 インデックス作成

インデックス	時間(分)	サイズ(GB)
単一	164.2	8.88
2 分割	65.2	9.34
4 分割	45.1	9.90

4.2 検索

NTCIR-6 CLIR における 140 題の検索課題(Topic)から 1,175 語を検索した際の検索時間を表 2 に示す．表 1 と同様に，分割インデックスの行には，分散並列により検索した際の最長時間を記載している．ここで，分割インデックスの検索時間を平均すると 2 分割では 36.7 秒，4 分割では 28.6 秒となっており，最長時間との差はあまりない．これは検索時にはディスク書き込みがほとんどなく，処理待ちが生じなかったためである．

分割インデックスを並列に検索することにより，検索時間を短縮することができた．本稿では分割インデックスの探索結果をマージしていないが，単純なマージ処理を行っても数秒程度（1.38 秒）であり，分割インデックスの方が速いと言える．

表 2 検索時間

インデックス	時間(秒)
単一	53.4
2 分割	39.0
4 分割	31.3

5. 考察

本稿で構築した，ログ構造の分割インデックスシステムを利用することにより，インデックスの作成，検索共に処理時間を短縮することができた．今後は，本分割インデックスシステムに更新処理機能を設け，その評価を行う必要がある．

また本稿において，分割インデックスを作成する際，処理待ちにより結果の出力時間に差が生じ，システム性能に影響を及ぼした．そのため，プログラム処理などにより，その差を短縮することができれば，分割インデックスシステムの更なる性能向上を目指すことができる．

6. おわりに

大規模文書集合の特性と最近のコンピュータの構成に適合するように，作成順の文書グループごとに小インデックスを作成する分割されたインデックスシステムをログ構造の分割インデックスと名付け，その構築を行った．

インデックスの構造と作成，検索および更新方法を提案した上で，実際に NTCIR-6 の文書データからインデックスを作成することにより，作成と検索性能の測定を行い，その有効性を確認した．

参考文献

[1]佐藤隆士：ログ構造を有する分割インデックス，電子情報通信学会第 18 回データ工学ワークショップ(DEWS2007), C1-2(2007-02).

[2]M.Rosenblum and J.K.Ousterhout : The Design and Implementation of a Log-structured File System, *ACM Trans . Computer System*, Vol. 10. No. 1, pp.26-52(1992).

[3]NTCIR-6 : http://research.nii.ac.jp/ntcir/ntcir-ws6/ws-ja.html.

[4]T.Sato, K.Han : NTCIR-3 CLIR Experiments at Osaka Kyoiku University –Compression of Gram-based Indices-, *Proc. NTCIR-3 CLIR*, pp.149-151(2002).

[5]Berkeley DB | Oracle Embedded Database : http://www.oracle.com/database/berkeley-db/index.html.