線形閉ハッシュ空間管理

奈良田　慧†　　三浦　孝夫†

† 法政大学 工学研究科　〒 184–8584 東京都小金井市梶野町 3-7-2
E-mail: †satoshi.narata.ze@stu.hosei.ac.jp, ††miurat@k.hosei.ac.jp

あらまし　動的ハッシュ法 (Dynamic Hash, DH) を用いる事によって, データ量に応じて空間サイズを動的に変化させ, 効率的に空間を獲得する事ができる. その代表的な技法には線形ハッシュ法 (Linear Hash, LH) が挙げられるが, 線形ハッシュ法ではあふれデータの偏りが解消できない問題と, 連続挿入によって分割が頻発してしまう問題がある. そこで, 本研究では線形閉ハッシュ法を提案する. あふれ領域を削除して, プライマリ領域だけでデータを管理し, あふれデータの偏りの軽減とスペース効率の向上を目指し, 従来の線形ハッシュ法と性能比較検証する.

キーワード　動的ハッシュ法 (Dynamic Hash, DH) , 線形ハッシュ法 (Linear Hash, LH)

Space Management at Linear Closed Hash

Satoshi NARATA† and Takao MIURA†

† Dept.of Elect.& Elect. Engr., HOSEI University　3-7-2, KajinoCho, Koganei, Tokyo, 184–8584 Japan
E-mail: †satoshi.narata.ze@stu.hosei.ac.jp, ††miurat@k.hosei.ac.jp

Abstract　In this investigation, we propose a new approach for bulk insert under linear hash organization, which is one of the known problems of dynamic hash techniques. Generally *Dynamic Hash* allows us to adjust the size of hash space dynamically according to the amount of data so that we obtain the nice time/space efficiency of the hash space. One of the typical techniques is *linear hash* where we can keep the hash space size linearly in terms of the amount of data. However, the technique doesn't always provide us with suitable properties, especially we face to severe deficiencies at bulk insert/delete operations, because the successive operations cause heavy manipulation (one extension at each operation) so that huge amount of I/O access happen to the secondary storage devices, called *thrashing*. There are other problems which cannot solve the deviation of overflow data. Then, We propose liner closed hash organization. It manages all data without an overflow Area. We compare with *linear hash* ,aiming at improvement in search efficiency and space efficiency.

Key words　Dynamic Hash,Linear Hash

1. 前　書　き

ハッシュ技法は O(1) で検索更新を行うことができ, オンライン実時間環境において有用である. しかしあふれ (衝突) やハッシュ関数の選択, ハッシュ空間の固定といった問題点が残されている. 動的ハッシュ法 (Dynamic Hash, DH) はこれらの問題のうちハッシュ空間の改善を目的としている. 線形ハッシュ法 (Linear Hash, LH) はこの代表的手法である. 線形ハッシュ技法はハッシュ空間サイズが格納データ量に応じて線形に増加するする機能を提供し, 時間計算量・空間量ともに安定した状態を保つものとして極めて優れた構造を有する [1]. 反面, 連続挿入によって分割が頻発し, 大量一括挿入では大幅に性能が低下することが知られ, 線形性の改善が提案されている [2]. この問題を解決するため, 著者らは予め推定した空間サイズにハッシュファイル構造を拡張し, 配置し直したデータをこの空間に格納する手法を提案した [3]. しかし,I/O 回数は 40%程度削減された反面, データ密度は 40%-65%と悪化している. そこで本研究では, あふれ領域を削除して, プライマリ領域だけでデータを管理し, 新たに局所閾値を設け不必要な分割処理を削減して, あふれデータの偏りの軽減とスペース効率の向上を目指す. 本稿では, 線形閉ハッシュ法を提案し, 線形ハッシュ法による個別拡張方式と比較しながら実験によりその有用性を示す.

2 章では,「線形ハッシュ法」について説明する. 3 章では, 提案手法である「線形閉ハッシュ法」について説明する. 4 章では, 実験とその結果による有用性の評価・考察を示し, 最後に第 5 章で結論とする.

2. 線形ハッシュ法

2.1 線形ハッシュ法アルゴリズム

よく知られているようにハッシュ技法は, あるキー値 X に対応するバケツをハッシュ関数 H を用いて指定する. ここでは, ハッシュ空間の格納領域の一つ一つを「バケツ」と呼ぶと仮定する. このバケツは直接アクセスでき, バケツ集合はハッシュ空間を構成する. 線形ハッシュ関数は関数 H に加え, 2つの非負整数パラメータ L,P で制御され, L はハッシュ空間の大域レベルを示し, P はバケツ成長点を示す. レベルはハッシュされたキー値のビット長を示しており, そのときのハッシュ関数を $H[L]$, $H[L+1](X)$ と $H[L](X)$ は下 L ビットが同じとする. 本稿では,

$$H[L](X) = X mod\ 2^{L} \tag{1}$$

とする.(1) の様な数式は一般的に,「ハッシュ関数」と呼ばれるものである. バケツは一定の条件下において2つに分割される. 一つはバケツが挿入によりあふれた場合, もう一つはハッシュ空間のデータ密度が大域閾値を超えた場合とする. データ密度は, バケツデータ総数に対する実際の格納データ数の割合で表される. 前者はあふれ域にチェインするとする. 後者の場合, バケツ成長値 P はこのとき分割されるバケツを示す. バケツ成長値を境に分割後ならばレベル $L+1$ を用い, 分割前なら L を用いてハッシュ計算を行う. 挿入の場合は当該バケツにデータを保存する. P の成長によりすべてのバケツが分割されると, レベル L が上昇し, 再び $P=0$ バケツから分割が始まる. 線形ハッシュ法の詳細については [1] を参照されたい. 以下に線形ハッシュ法についての例題を示す.

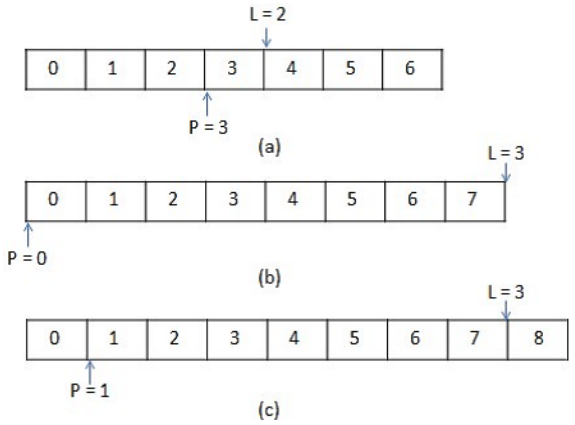

図 1　線形ハッシュ法

図 1(a) ではバケツ番号 0,1,2 まで分割されており, バケツ番号 4,5,6 がこのハッシュ空間レベル L での分割によって拡張したハッシュ空間である. この図 1(a) の状態に一件のデータ挿入があり, それによってデータ密度が大域閾値を越え分割が起こったと仮定する. その結果が図 1(b) である. 分割の際 $P=2^{L}$ に達した,L のレベルが 1 上昇し, 再び $P=0$ からとなる. 図 1(a) では分割の結果, 次は $P=4$ となるが $P=2^{L}(L=2)$ に達したので, 図 1(b) の様に L のレベルが 1 上昇し, 再び $P=0$ になっている. また図 1(b) の状態に一件のデータ挿入があり, それによってデータ密度が大域閾値を超え分割が起こったと仮定する. その結果が図 1(c) であり,$P=1$ となりバケツ 8 が拡張している. この様に, 線形ハッシュはハッシュ空間サイズを線形に増加させてゆく.

2.2 線形ハッシュ法の利点

線形ハッシュ構造の利点は, 大域閾値を設けることによってあふれ (衝突) を減少させれる事にある. また, ハッシュ空間サイズが線形に拡張するので, あふれ (衝突) 確率が安定しており, 予め膨大なハッシュ空間を確保する必要性が無い.

2.3 線形ハッシュ法の欠点

線形ハッシュ構造の欠点は, バケツ成長位置だけが分割対象なため, 例え他にあふれデータを多く保持しているバケツがあるとしても, そのバケツに成長位置ポインタ P の順番が回ってこないと, あふれデータを解消する機会が無い事にある. また, 一旦大域閾値を超えたデータ密度は大きく低下しないため, 連続挿入で線形拡張による入出力スラッシングが頻発する事がある. 以下にこの問題についての例題を示す.

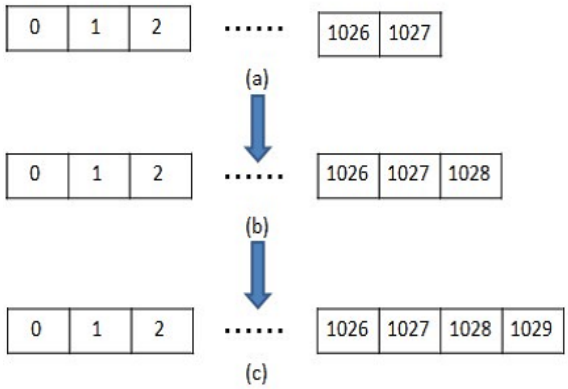

図 2　線形ハッシュ法の欠点

図 2(a) はバケツ数 1028 個からなる大域閾値 0.8 のハッシュファイルであり, 図 2(a) には 822 個データが入っておりデータ密度は 79.96%であると仮定する. この図 2(a) の状態にデータが一件挿入され, 図 2(a) に 823 個のデータが存在すると仮定する. このとき, データ密度は 80.07%に達するため, 大域閾値を超えたため分割が発生し, 図 2(b) の状態に移行する. 分割が発生しハッシュ空間は増加したため, データ密度は低下して 79.98%となる. この図 2(b) の状態にデータが一件挿入され, 図 2(b) に 824 個のデータが存在すると仮定する. このとき, データ密度は 80.08%に達するため, 大域閾値を超えたため分割が発生し, 図 2(c) の状態に移行する. この様に, データ密度が大域閾値付近で停滞している事が原因で, 一件の挿入の度に空間の分割が発生する現象が起こりやすく,I/O が急激に増加する要因になっている.

3. 線形閉ハッシュ法

3.1 線形閉ハッシュ法の狙い

線形ハッシュ構造の欠点は前述のとおり, バケツ成長位置だけが分割されてあふれデータを多く持つバケツの問題が解消されず, また一旦大域閾値を超えたデータ密度は大きくは低下しないため, 連続挿入で分割が頻発することにある. そこで, プライマリ領域だけでデータを管理し, あふれデータを一時的に保持する空きバケツを予め一定数以上確保する機構を実装する. またバケツ成長時だけでなく, データ挿入時に際しても対象バケツに対して分割処理判定を行い, 局所閾値以上のレコード数を対象バケツが保持してたら分割処理を実行する事によって, 線形ハッシュ法に比べ検索効率とスペース効率の向上が図れると予想される.

3.2 バケツ管理

線形閉ハッシュ法では, 新たに以下のパラメータを定義する. l(あふれ $l = -1$, 空きバケツ $l = -2, L \geqq l$) はハッシュ空間の各バケツ毎の局所レベルを示す. thr は各バケツ毎のデータ密度の局所閾値を示す. $Previous, Next$(どのバケツも指してない $= -1$) はバケツ同士の相互ポインタを示す. $FreeList$ はあふれデータを一時的に保持する空きバケツを管理し, 一定数以下になったらある閾値まで $Last$ の後ろから確保する. $Last(P \leqq Last)$ はハッシュ空間の最後のバケツを指し,$P = Last$ に達したら $Last$ のバケツを FreeList に空きバケツとして登録し,1 増加させる.

3.3 検　　索

線形閉ハッシュ法の検索操作手順を論じる. 検索したいデータを大域レベル L のハッシュ関数でハッシュ し, その結果のハッシュ値が P 未満ならば, ハッシュ関数のレベルを 1 増加させて再ハッシュを行う. 当該バケツが溢れとして使われているか空きである ($l < 0$ かどうか) 限り, ハッシュ関数のレベルを 1 減少させ再ハッシュを行う. この操作をプライマリとして使われてるバケツ ($l \geqq 0$) を発見するまで続ける. 当該バケツを発見したら該当バケツ及びそのあふれを探索 する (図 3).

以下に線形閉ハッシュ法の検索操作ついての例題を示す.

図 3 は大域レベル $L = 3$, 大域閾値 $THR = 0.90$, 局所閾値 $thr = 0.80$, バケツ成長点 $P = 1$ の時のプライマリ領域を表している. 以後, プライマリ領域の 1 バケツあたり 5 レコードまで格納できるものとする. この状態からデータ "X "の検索を行う. データ "X "のハッシュ値を 2 と仮定する. $2 \geqq 0$ なのでバケツ番号 2 の局所レベル l を確認する. $l \geqq 0$ だと仮定するとバケツ番号 2 番とそれにチェインされているバケツ番号の探索を行う. その結果, データ "X "が含まれているかが判明する.

3.4 挿　　入

線形閉ハッシュ法の挿入操作手順を論じる. 挿入したいデータを大域レベル L のハッシュ関数でハッシュ し, その結果のハッシュ値が P 未満ならば, ハッシュ関数のレベルを 1 増加させて再ハッシュを行う. 当該バケツが溢れとして使われているか空きである ($l < 0$ かどうか) 限り, ハッシュ関数のレベルを 1 減少させ再ハッシュを行う. この操作をプライマリとして使われてるバケツ ($l \geqq 0$) を発見するまで続ける. 当該バケツを発見したら該当バケツ又はそのあふれに挿入する (図 4). 該当バケツが既に一杯で, チェインされているあふれバケツが無いとき, 事前に確保してある空きバケツにあふれデータを挿入する (図 5).

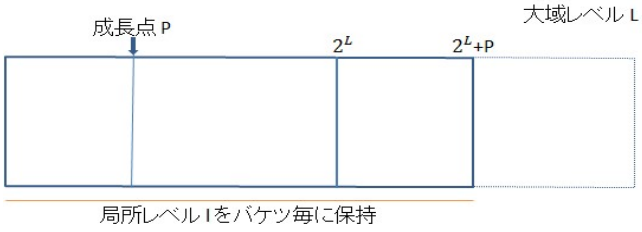

図 3　線形閉ハッシュ法検索操作

以下に線形閉ハッシュ法の検索操作ついての例題を示す.

図 4 は大域レベル $L = 3$, 大域閾値 $THR = 0.90$, 局所閾値 $thr = 0.80$, バケツ成長点 $P = 3$ の時プライマリ領域を表している. この状態からデータ "Y "の挿入を行う. データ ""Y のハッシュ値を 0 と仮定する. $0 < P$ なのでレベル L+1 で再ハッシュしたハッシュ値が 4 だと仮定する. バケツ番号 4 の局所レベル l を確認する.$l \geqq 0$ だと仮定するとバケツ番号 4 番に挿入を試みる. しかし, バケツ番号 4 番いっぱいで挿入できなかった為, 図 5 のように事前に確保してあった空のバケツにデータ "Y "を挿入し, チェインをして局所レベル l をあふれを示す-1 に書き換える.

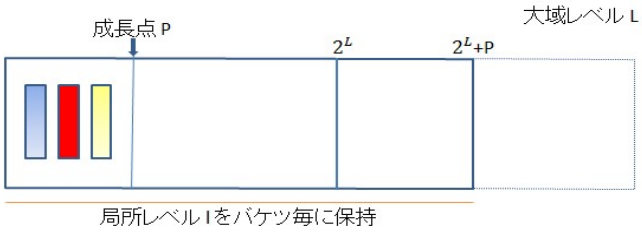

図 4　線形閉ハッシュ法挿入操作前

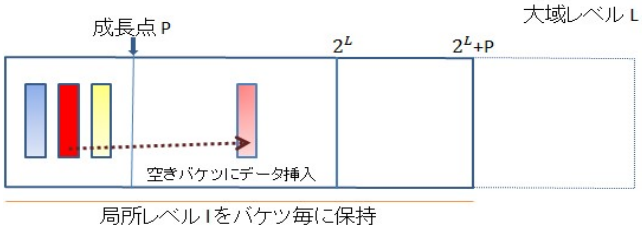

図 5　線形閉ハッシュ法挿入操作後

3.5 分割処理

線形閉ハッシュ法のバケツ成長点の増加による分割処理手順を論じる．データ密度が大域閾値を超えて P が成長する時, 該当バケツ内のデータ密度が局所閾値を超えていたら分割可能か判定処理を行う．対象バケツの密度が局所閾値未満だった場合や局所閾値以上でも再ハッシュ時,1 通りのハッシュ値しかないならば分割処理は行われず,P が成長するだけとなる (図 6). 分割該当先バケツがあふれバケツとして既に使われている場合は, 事前に確保してある空きバケツにデータを移して空にする必要がある．分割成功後, 該当バケツの局所レベルを再ハッシュしたレベルに更新する (図 7).

以下に線形閉ハッシュ法のバケツ成長点の増加による分割処理ついての例題を示す.

図 6 は大域レベル $L = 3$, 大域閾値 $THR = 0.90$, 局所閾値 $thr = 0.80$, バケツ成長点 $P = 3$ のプライマリ領域を表している．データ ”Z ”の挿入直後と仮定し, データ密度が大域閾値を超過し, 成長点 P の増加によりバケツ番号 P が再ハッシュ対象となる．しかしバケツ番号 P のバケツ密度が局所閾値を下回った為, 分割は発生せず P が成長するだけとなる．図 7 は大域レベル $L = 3$, 大域閾値 $THR = 0.90$, 局所閾値 $thr = 0.80$, バケツ成長点 $P = 3$ のプライマリ領域を表している．データ ”α ”の挿入直後と仮定し, データ密度が大域閾値を超過し, 成長点 P の増加によりバケツ番号 P が再ハッシュ対象となる．バケツ番号 P のバケツ密度が局所閾値を上回った為分割が発生したが, 分割該当先バケツがあふれバケツとして使われていた為, 事前に確保してあった空のバケツにあふれデータを移動させてから分割を行い, 分割該当先バケツの局所レベルを再ハッシュしたレベルに更新した.

線形閉ハッシュ法の挿入操作による分割処理手順を論じる．データ挿入時, バケツ毎密度が局所閾値を超過かつ $L \geqq l$ かつ P 未満の場合又は挿入該当バケツのバケツ毎密度が局所閾値を超過かつ $L > l$ の場合, どちらかを満たす場合該当バケツに対して分割可能か判定処理を行う．対象バケツの密度が局所閾値未満だった場合や局所閾値以上でも再ハッシュ時,1 通りのハッシュ値しかないならば分割処理は行われない．分割該当先バケツがあふれバケツとして既に使われている場合は, 事前に確保してある空きバケツにデータを移して空にする必要がある．分割成功後, 該当バケツの局所レベルを再ハッシュしたレベルに更新する (図 8).

以下に線形閉ハッシュ法の挿入操作による分割処理についての例題を示す.

図 8 は大域レベル $L = 3$, 大域閾値 $THR = 0.90$, 局所閾値 $thr = 0.80$, バケツ成長点 $P = 3$ のプライマリ領域を表している．データ ”γ ”の挿入直後と仮定し, データを挿入したバケツのデータ密度が局所閾値を超過かつ $L \geqq l$ かつ P 未満を満たしたので分割処理が発生したとする．しかし, 分割該当先バケツがあふれバケツとして使われていた為, 事前に確保してあった空のバケツにあふれデータを移動させてから分割を行い, 分割該当先バケツの局所レベルを再ハッシュしたレベルに更新した.

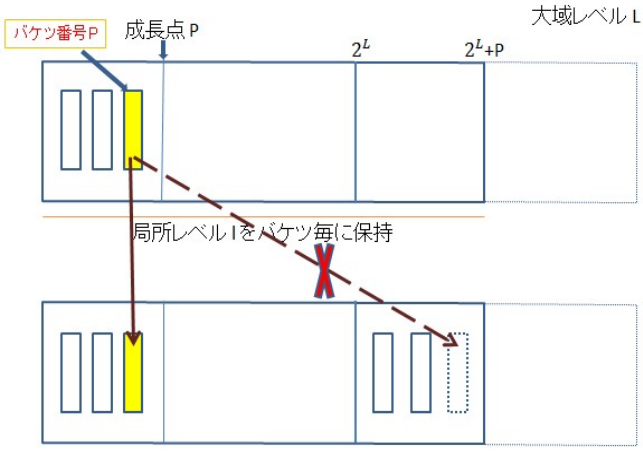

図 6　線形閉ハッシュ法のバケツ成長点の増加による分割処理せず

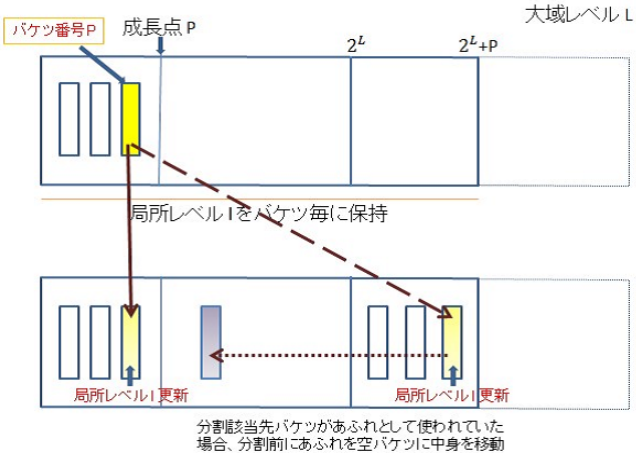

図 7　線形閉ハッシュ法のバケツ成長点の増加による分割処理後

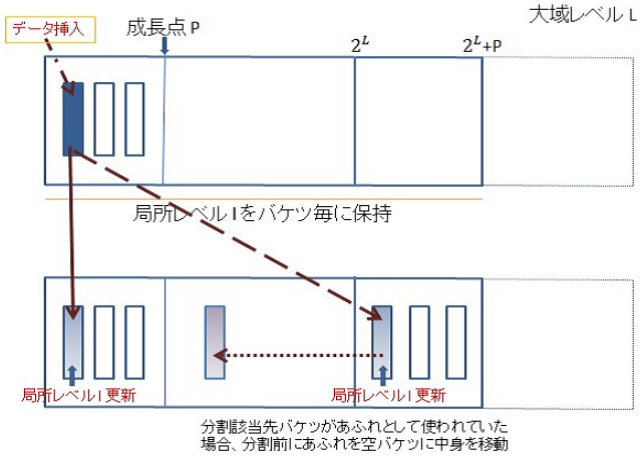

図 8　線形閉ハッシュ法の挿入操作による分割処理後

4. 実　　験

4.1 準　　備

本章では, 線形閉ハッシュ法の有用性を検証するために, 性能調査や線形ハッシュ技法 (Linear Hash,LH) との比較検証を行う. 本研究では, 挿入・検索に用いるデータ郡として, 非一様分布データであるフィラデルフィア周辺の郵便アドレス(注1) 5000 件,10000 件,50000 件,60000 件を用いる. また,MersenneTwister (注2) を用いて,0 から 1 の一様分布データ 5000 件,10000 件,50000 件,60000 件を作成し, 用いる. プログラム作成には Java 言語を用いる. 評価対象要因は, プライマリ域読み込み回数, プライマリ域書き込み回数あふれ域読み込み回数, あふれ域書き込み回数, あふれデータ数, 探索成功率 50%の 10000 件のバケツタッチ数, レベル L, バケツ成長点 P も一緒に記録するものとする. また, 比較条件としては次の５つを考える.

（1） 線形ハッシュ法での実験において, 閾値は 0.7, 0.8, 0.9, 0.95 の各々について行う.

（2） 1 バケツあたり 5 レコード格納可能とする.

（3） 線形ハッシュ法ではあふれデータは個別線形リストで管理する.

（4） 線形閉ハッシュ法ではあふれデータもプライマリ領域で管理する.

（5） 線形閉ハッシュ法の各バケツの局所閾値を 0.8 とする.

4.2 実験項目

実験内容は以下の項目について行う.

- 線形ハッシュ法・線形閉ハッシュ法で行う実験

①5000 件,10000 件,50000 件,600000 件の新規作成
②50000 件のハッシュデータに対する 5000 件,
10000 件,50000 件,60000 件の追加挿入

4.3 実験目的

4.2 の ①,② から累計読み込み回数, 累計書き込み回数, 探索成功率 50%の 10000 件のバケツタッチ数, あふれデータ数の性能比較を行い考察する.

4.4 実験結果

(注1)：本実験では,www.rtreeportal.org で公開されている地図情報を用いる.
(注2)：MersenneTwister についての詳細アルゴリズムは www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/mt.html にて紹介されている.

閾値	新規作成件数	プラ読み回数	プラ書き回数	あふれ読み回数	あふれ書き回数	成功率 50 %探索 10000 件バケツタッチ数	あふれデータ数	L	P
0.7	5000 件	8603	5174	6917	3501	14426	696	10	206
0.7	10000 件	17208	10346	13863	7013	13529	1398	11	410
0.7	50000 件	84686	50661	68106	34671	13867	8496	13	3667
0.7	60000 件	102689	61132	84212	42485	13434	9037	13	6369
0.8	5000 件	8087	4042	8575	4180	15340	834	10	18
0.8	10000 件	16202	8072	17297	8402	14370	1637	11	43
0.8	50000 件	78735	38933	82538	40925	15304	11205	13	1507
0.8	60000 件	92926	45170	98564	49037	16028	15526	13	2927
0.9	5000 件	7006	2368	10286	4786	23052	1970	9	162
0.9	10000 件	14125	4790	20819	9649	20545	3778	10	359
0.9	50000 件	74568	26399	110135	50164	16578	13029	13	24
0.9	60000 件	86823	30206	126651	58681	18116	19600	13	786
0.95	5000 件	6702	1696	12035	5181	25184	2291	9	59
0.95	10000 件	13443	3404	24240	10410	23511	4531	10	128
0.95	50000 件	66069	15768	121736	51973	23055	24503	12	1272
0.95	60000 件	84614	22577	154385	64839	19395	20972	13	25

表 1　一様分布データでの線形ハッシュ法新規作成

閾値	新規作成件数	プラ読み回数	プラ書き回数	あふれ読み回数	あふれ書き回数	成功率 50 %探索 10000 件バケツタッチ数	あふれデータ数	L	P
0.7	5000 件	8590	5163	6954	3503	14464	708	10	203
0.7	10000 件	17177	10370	13732	6940	13515	1403	11	409
0.7	50000 件	84122	49676	69163	35162	14122	9141	13	3483
0.7	60000 件	101178	59077	85301	43044	13904	10754	13	5879
0.8	5000 件	8065	3951	8835	4258	15415	867	10	10
0.8	10000 件	16136	7989	17304	8409	14611	1725	11	21
0.8	50000 件	78329	38192	83724	41335	15425	11704	13	1383
0.8	60000 件	92161	43918	100180	49592	16328	16513	13	2680
0.9	5000 件	7048	2346	10697	4874	22741	1908	9	176
0.9	10000 件	14027	4691	21026	9703	20641	3863	10	340
0.9	50000 件	73237	25337	109676	49792	17291	15011	12	3679
0.9	60000 件	86457	29658	128156	58915	18143	20132	13	668
0.95	5000 件	6695	1648	12573	5243	25488	2303	9	56
0.95	10000 件	13472	3358	24653	10490	23166	4487	10	137
0.95	50000 件	66013	15678	122257	52088	22792	24602	12	1251
0.95	60000 件	82102	20726	152755	63907	20895	24950	12	3283

表 2　非一様分布データでの線形ハッシュ法新規作成

閾値	新規作成件数	プラ読み回数	プラ書き回数	成功率 50 %探索 10000 件バケツタッチ数	あふれデータ数	L	P
0.7	5000 件	7740	8187	11074	316	10	851
0.7	10000 件	15860	16964	10734	570	11	1794
0.7	50000 件	78404	81853	10726	3162	14	626
0.7	60000 件	95494	100170	10789	4452	14	4732
0.8	5000 件	9483	9002	11855	579	10	428
0.8	10000 件	19505	18653	11298	1075	11	920
0.8	50000 件	101534	96683	11279	5181	13	6850
0.8	60000 件	120893	115002	11160	6102	14	581
0.9	5000 件	10292	9146	12375	728	10	177
0.9	10000 件	20705	18424	11749	1449	11	402
0.9	50000 件	106929	95395	11879	7968	13	4311
0.9	60000 件	129880	115616	11853	8945	13	6933
0.95	5000 件	10430	8952	12657	800	10	102
0.95	10000 件	20948	17997	11905	1575	11	204
0.95	50000 件	108014	93265	12134	9040	13	3410
0.95	60000 件	131229	113103	12208	10664	13	5861

表 3　一様分布データでの線形閉ハッシュ法新規作成

閾値	新規作成件数	プラ読み回数	プラ書き回数	成功率 50 %探索 10000 件バケツタッチ数	あふれデータ数	L	P
0.7	5000 件	8154	8460	10993	312	10	934
0.7	10000 件	16330	17252	10788	622	11	1649
0.7	50000 件	82384	85810	10747	3554	14	949
0.7	60000 件	100378	105251	10845	4909	14	5492
0.8	5000 件	9989	9483	11870	553	10	431
0.8	10000 件	20209	19288	11279	1082	11	870
0.8	50000 件	103666	98622	11330	5511	13	7135
0.8	60000 件	123147	116808	11258	6845	14	795
0.9	5000 件	10435	9319	12417	733	10	188
0.9	10000 件	21294	19026	11747	1461	11	402
0.9	50000 件	109496	97701	11912	8187	13	4485
0.9	60000 件	132641	117943	11878	9416	13	7161
0.95	5000 件	10578	9113	12630	797	10	107
0.95	10000 件	21756	18842	11925	1603	11	214
0.95	50000 件	112395	97364	12200	9322	13	3546
0.95	60000 件	135854	117206	12215	11133	13	6059

表 4　非一様分布データでの線形閉ハッシュ法新規作成

閾値	追加挿入件数	プラ読み回数	プラ書き回数	あふれ読み回数	あふれ書き回数	成功率 50 %探索 10000 件バケツタッチ数	あふれデータ数	L	P
0.7	5000 件	8630	4954	7633	3772	13818	9197	13	4895
0.7	10000 件	17900	10380	15986	7786	13487	9147	13	6338
0.7	50000 件	83421	48393	70725	35740	14249	18446	14	6918
0.7	60000 件	100409	57807	86616	43487	14169	20153	14	9287
0.8	5000 件	7125	3175	7908	4031	15712	13322	13	2228
0.8	10000 件	14197	6192	16156	8152	16046	15528	13	2927
0.8	50000 件	77532	36770	85656	42022	15702	23991	14	2619
0.8	60000 件	91377	42670	101566	50060	16206	28771	14	3924
0.9	5000 件	6153	2017	7973	4168	17379	16273	13	415
0.9	10000 件	12199	3740	16556	8526	18145	19682	13	768
0.9	50000 件	68502	21540	105930	48458	18720	35142	13	6221
0.9	60000 件	85589	28530	130150	59311	17789	34477	14	399
0.95	5000 件	7995	2530	14904	5833	22137	24757	12	2271
0.95	10000 件	18535	32611	12823	19452	20986	20986	13	22
0.95	50000 件	65130	14929	121132	51902	23117	50488	13	2232
0.95	60000 件	78683	18144	148654	62650	23135	54855	13	3418

表 5　一様分布データでの線形ハッシュ法 50000 件に対する追加挿入

閾値	追加挿入件数	プラ読み回数	プラ書き回数	あふれ読み回数	あふれ書き回数	成功率 50 %探索 10000 件バケツタッチ数	あふれデータ数	L	P
0.7	5000 件	8298	4551	7756	3859	14170	10208	13	4606
0.7	10000 件	17353	9798	16038	7802	13840	10407	13	5978
0.7	50000 件	82050	45911	73624	36929	14357	20639	14	6291
0.7	60000 件	97977	54199	89613	44771	14439	23533	14	8321
0.8	5000 件	6923	2997	7848	4007	15954	14091	13	2036
0.8	10000 件	14045	6055	16247	8155	16268	16225	13	2752
0.8	50000 件	76407	34910	88350	42815	15741	26001	14	2116
0.8	60000 件	89889	40373	104722	50955	16216	31237	14	3307
0.9	5000 件	7175	2666	9648	4695	17409	16729	13	313
0.9	10000 件	13324	4486	18363	9070	18125	19969	13	704
0.9	50000 件	66498	19669	106379	48165	19566	40125	13	5114
0.9	60000 件	85479	27548	136328	60407	17701	36630	13	8112
0.95	5000 件	7187	1926	14198	5524	22496	26136	12	1981
0.95	10000 件	16748	5551	31232	12081	20510	23900	12	3502
0.95	50000 件	64261	14125	122748	51760	23327	51967	13	1920
0.95	60000 件	76499	16307	149925	62106	23671	58420	13	2667

表 6　非一様分布データでの線形ハッシュ法 50000 件に対する追加挿入

閾値	追加挿入件数	プラ読み回数	プラ書き回数	成功率 50 %探索 10000 件バケツタッチ数	あふれデータ数	L	P
0.7	5000 件	8781	9138	10732	3729	14	2363
0.7	10000 件	17065	18299	10816	4431	14	4703
0.7	50000 件	79361	82445	10767	6562	15	1720
0.7	60000 件	97475	101722	10835	7992	15	5733
0.8	5000 件	10885	10246	11166	5211	14	13
0.8	10000 件	20927	19890	11191	6071	14	570
0.8	50000 件	97095	91642	11362	10501	14	14239
0.8	60000 件	113518	106360	11239	10729	15	47
0.9	5000 件	11437	10155	11885	8616	13	5656
0.9	10000 件	22911	20217	11883	8971	13	6930
0.9	50000 件	112384	99665	11993	16337	14	9047
0.9	60000 件	135433	119817	11976	17354	14	11792
0.95	5000 件	11504	9846	12210	9988	13	4664
0.95	10000 件	23198	19858	12203	10663	13	5868
0.95	50000 件	114031	98144	12318	18667	14	7094
0.95	60000 件	137392	117910	12312	20402	14	9649

表 7　一様分布データでの線形閉ハッシュ法 50000 件に対する追加挿入

閾値	追加挿入件数	プラ読み回数	プラ書き回数	成功率 50 %探索 10000 件バケツタッチ数	あふれデータ数	L	P
0.7	5000 件	9181	9632	10803	4270	14	2911
0.7	10000 件	17808	19314	10817	4807	14	5409
0.7	50000 件	85794	88678	10795	8091	15	3256
0.7	60000 件	104684	109593	10846	9710	15	8644
0.8	5000 件	9083	8306	11255	5805	14	20
0.8	10000 件	19321	18032	11224	6720	14	722
0.8	50000 件	100232	93967	11313	11630	14	15004
0.8	60000 件	119369	110766	11236	12705	15	253
0.9	5000 件	11515	10114	11961	8934	13	5850
0.9	10000 件	22968	20163	11872	9322	13	7160
0.9	50000 件	114370	100437	11973	17463	14	9457
0.9	60000 件	137925	120831	11924	18573	14	12286
0.95	5000 件	11654	9965	12223	10270	13	4821
0.95	10000 件	23274	19810	12216	10984	13	6060
0.95	50000 件	115991	98835	12262	19798	14	7399
0.95	60000 件	139956	118924	12262	21626	14	10092

表 8　非一様分布データでの線形閉ハッシュ法 50000 件に対する追加挿入

4.5 考　　察

最初に読み込み書き込み回数の比較を行う．表 1,2,3,4 の閾値 0.95,60000 件の新規作成の読み込み書き込み回数を比較すると, 線形閉ハッシュ法は線形ハッシュ法に比べ, 読み込み回数が平均 0.57 倍と改善している反面, 書き込み回数が平均 1.34 倍と悪化している．表 5,6,7,8 の閾値 0.95,50000 件のデータに対する 60000 件の追加挿入での読み込み書き込み回数を比較すると, 線形閉ハッシュ法は線形ハッシュ法に比べ, 読み込み回数が平均 0.62 倍と改善している反面, 書き込み回数が平均 1.28 倍と悪化している．

次に探索成功率 50%の 10000 件のバケツタッチ数の比較を行う．表 1,2,3,4 の閾値 0.9・0.95,5000 件,10000 件,50000 件,60000 件の新規作成での探索成功率 50%の 10000 件のバケツタッチ数を比較すると, 線形閉ハッシュ法は平均 12105 回のバケチタッチ数だが, 線形ハッシュ法では平均 212871 回のバケツタッチ数となっており, 平均タッチ数 0.57 倍と高速化に成功していることが確認できる．表 5,6,7,8 の閾値 0.9・0.95,50000 件のデータに対する 5000 件,10000 件,50000 件,60000 件の追加挿入での探索成功率 50%の 10000 件のバケツタッチ数を比較すると, 線形閉ハッシュ法は平均 12092 回のバケチタッチ数だが, 線形ハッシュ法では平均 20263 回のバケチタッチ数となっており, こちらも平均タッチ数 0.60 倍と高速化に成功していることが確認できる．

最後にあふれデータ数について比較を行う．表 1,2,3,4 の閾値 0.9・0.95,50000 件,60000 件の新規作成でのあふれデータ数の比較をすると, 線形閉ハッシュ法は線形ハッシュ法に比べ平均 0.46 倍, 最大で 0.37 倍に削減している事が確認できる．表 5,6,7,8 の閾値 0.9・0.95,50000 件のデータに対する 50000 件,60000 件の追加挿入でのあふれデータ数の比較をすると, 線形閉ハッシュ法は線形ハッシュ法に比べ平均 0.42 倍, 最大で 0.37 倍に削減している事が確認できる．

以上の実験値より, 線形閉ハッシュ法ではデータの頻繁な移動に伴う書き込み回数増加の結果, あふれデータ数が減少し, それに伴い探索時のバケツタッチ回数の軽減に起因したと推測できる．そして, 線形ハッシュ法では膨大なあふれデータ数が探索時のバケツタッチ数を悪化させたと推測できる．また全ての表から, 線形閉ハッシュ法では探索時のバケツタッチ回数がデータ数ではなく閾値でほぼ一定となっていたため, 閾値依存と推測できる．

5. 結　　論

本研究では, 書き込み回数の増加を伴うが, 大幅な検索効率の向上が可能であると示した．また, 線形ハッシュ法ではあふれデータの増加による性能悪化を解消しにくいが, 線形閉ハッシュ法では改善を見込め, プライマリ領域だけでデータを効率よく管理でき, スペース効率の向上も見込める利点も示した．

文　　献

[1] Litwin, W.: Linear hashing - A New Tool for File and Table Addressing, VLDB, 1980

[2] Yasuda, K., et al.: Distributed Processes on Tree Hash, COMPSAC, 2006

[3] Narata,S,, et al.: On Inserting Bulk Data for Linear Hash Files, NDT, 2011

[4] Ashok Rathi, Huizhu Lu, G.E. Hedrick: Performance Comparison of Extendible Hashing and Linear Hashing Techniques, ACM SIGSMALL/PC Symposiumon Small Systems,1990

[5] Ellis, C.S.: Concurrency and Linear Hashing, PODS, 1985

[6] James K. Mullin: Tightly controlled linear hashing without separate overflow storage, BIT Numerical Mathematics Volume 21-4, pp.390-400

[7] Buckhard, W.A.: Associative Retrieval Trie Hash-Coding, Journal of Computer and System Sciences (JCSS) 15, pp.280-299 (1977)