Vol.2014-MPS-97 No.9
2014/3/3

2 段階重複排除の効果の分析

山崎寛人 †　小池到 †　木下俊之 †

企業などの大規模データストレージでは，保持するデータ量が年々増加している．この大容量のストレージ内には，同一または類似したファイルが増え，無駄なデータ領域を消費する傾向にある．そこで注目されているのが，データ重複排除技術である．重複排除技術は、ファイルの重複部分をひとつにまとめてファイル量を削減する技術である。2 段階重複排除は重複排除を 2 段階に分けることで、データ削減効果を維持しながら重複排除の処理時間の短縮を図る。本研究は、この 2 段階重複排除のデータ削減効果と処理時間の短縮特性を明らかにした。その結果、2 段階にすることによりメタデータを含むデータの削減率は 3%前後低下するが、処理時間は 20%から 35%程度短縮する結果となった。

Analysis of double layered deduplication efficiency

Hiroto Yamasaki†　Itaru Koike†　Toshiyuki Kinoshita†

Nowadays, volumes of data storage in enterprise systems keep increasing and led to large scale of data storage. Mostly or exactly identical and similar multiple versions of files are increasingly being saved in data storage and it lead to the wasteful space in the data storage. This problem could be resolved by using deduplication techniques - a powerful storage optimization technique that combined duplicated file and reduce the file volume by deleting same data. Double layer deduplication technique is a technique that divides deduplication to two layers and while reducing the data it also could maintaining and minimizing the time taken in performing the deduplication. This research proved that two layer deduplication techniques could reduce data while at the same time keeping the minimum processing time. Results that obtained by performing two layers deduplication shown that processing time was decreased in range of 20% until 35%. We have evaluated the effectiveness by experimental in real environment.

1. 序論

企業などの大規模データストレージでは，保持するデータ量が年々増加している. この大容量のストレージ内には，同一または類似したファイルが増え，データ領域を無駄に消費する傾向にある．企業が保持するデータ量の増加は今後も続くと考えられ，加えてビッグデータの分析・活用が行われるようになれば，さらに顕著なものになっていくと予想される．そこで注目されているのが，データ重複排除技術 (De-duplication，Dedup)である．データ重複排除技術とは，データ圧縮法の 1 つで，ディスクボリューム全体で重複する部分を探し出し，それを排除する技術である．

重複排除技術は大規模なデータストレージに対して，個々のファイルを圧縮する従来の方法よりも効率的に圧縮でき，保持するデータをより小さくすることができる．加えて, zip など従来のデータ圧縮法と併用することもできるため，高い圧縮効果を見込める[1]．しかし，重複排除技術は通常のデータ圧縮法よりも適用範囲が広いため処理が複雑になることに加えて, 類似性の低いデータ群の場合には，複雑な処理をする割には効果を上げられないという問題がある．また，ファイルサイズが小さい場合や，データの重複部分が小さい場合，データの重複を検出できず効果的な排除できないことがある．

† 東京工科大学大学院，コンピュータサイエンス専攻
Tokyo University of Technology Graduate School, computer science program

多段階重複排除は，データをユーザ側からバックアップサーバに送る過程で，複数回にわたって重複排除する方法である[2]. 従来はバックアップサーバのみで重複排除を行っていたため，重複排除にかかる負荷が一部に集中していた．だが，多段階重複排除であれば負荷を分散することができることに加えて，従来よりも処理時間やメタデータ量を削減することができる．

2. 重複排除

データ重複排除は，ファイルレベルまたはブロックレベルでデータの重複がないかを調べ，重複が見つかった場合は，初めに検出したオリジナルとなるデータのみをストレージに保存し，重複箇所にはそのデータを示すポインタを作成し，データ量を削減する．

重複排除では，2 回目以降に出現する重複データは，同データが初めて出現した位置を指し示すポインタに置き換えられるため，オリジナルのデータのみがストレージに保存される．ブロックレベルの重複判定では，データを複数のブロックに分割し，これらブロックのハッシュ値を計算する．このブロックはチャンクとも呼ばれ，データストリームから切り出し生成されたデータブロックのことを指す. 算出されたハッシュ値はフィンガープリント（FP）とも呼ばれ，それぞれのブロックを判別するための識別子として用いられる．これらの識別子は，ユニークな識別子のみで構成されたデータベースに対して照合される．識別子がデータベース内にすでに存在するときは，そのデータが重複

していることを意味するため，データベース内の該当するユニークな識別子を指し示すポインタ（リンクデータ）に置き換えられる．また，データベース内に一致する識別子が存在しなかったときは，ユニークな識別子と判断されて新しくデータベースに格納される．新しく認識されたユニークなブロックは，ポインタに置き換えることなくそのまま保存される．

たとえば，バックアップデータの総容量が 100 ギガバイトで，その内の 20 ギガバイトが重複データである場合，実際にストレージに保存されるのは 80 ギガバイトのユニークなデータであり，その後で通常のデータを圧縮すればさらにディスク容量を節約できる． 図 1 にこの重複排除の処理の概要を示す．

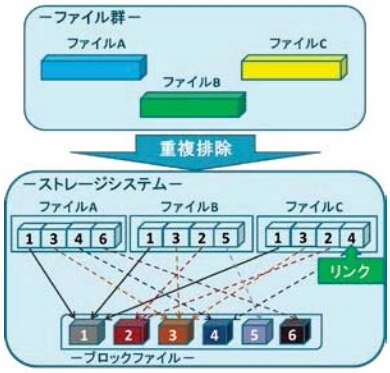

図 1　重複排除概要

2.1 重複排除の単位

重複排除における重複判定方式は，大きくファイル単位とブロック単位，バイト単位の 3 種類に分けられる．

ファイル単位の重複排除では，ファイルごとハッシュ値を算出し，重複の有無を走査する．この方式は，シングルインスタンスと呼ばれることもある．例えば同じ職場の複数の社員がファイルサーバ上に同じファイルをコピーするなど，重複ファイルがストレージ内に多く存在する場合は大きな効果を発揮する．ファイル単位の弱点は，ファイルに少しでも変更が入ると重複排除がまったく効かなくなる点にある．ファイルに更新が入る場合，多くの場合はファイルの一部だけが書き換えられて新しいバージョンが作られる．ファイル単位の重複排除では，こういったバージョン間の重複部分を認識できない．その結果，後に説明するブロック単位やバイト単位に比べて，重複排除効果が劣ることが欠点である．

ブロック単位の重複排除では，ファイルをいくつかの部分に区切ったブロックごとに重複の有無を走査する．ブロック単位の方式には，固定長ブロック方式と可変長ブロック方式の 2 つがある．

固定長ブロック方式は，ファイルを一定のサイズのブロックに分割し，そのブロックごとに重複判定を行う．この方式によるソフトウェアは，一般にブロックサイズは 4 キロバイトから 128 キロバイトに設定される．この範囲で重複排除の対象のファイルに合わせてユーザがブロックサイズを選択肢から選ぶことができる．ブロックサイズを選択できるようにする理由は，重複排除率と重複確認を行う際の処理速度にある．ブロックサイズが小さいほど重複排除率は向上するが，生成されるブロック数が増加してしまい，データベースに格納されている識別子と照合して重複確認を行うのに要する時間が延びてしまう．これにより重複排除を行う処理速度が低下する．逆に，ブロックサイズを大きくすると，生成されるブロック数は減少するので，重複確認にかかる時間は比較的短くなり処理速度は向上する．しかし，ブロックサイズを大きくするとデータが重複する確率は格段に低下し，重複排除率も低下する．このように重複排除率と処理速度はトレードオフの関係にあることから，重複排除を行う環境にあわせてブロックサイズを適切に決める必要がある．現在，オープンソースで公開されている重複排除ソフトにはこの方式が多く，lessfs[3]や ZFS[4]などに使われている．

可変長ブロック方式は，データパターンを見ながらブロックの切れ目を決める．ブロックという言葉は固定長のものに使われることが多いため，可変長ブロックのことをセグメントやチャンクとも呼ぶことがある．

ブロックの生成は，始めに最小サイズのブロックであるオフセットと，それに加えて一定サイズのウィンドウを読み込み，これをハッシュ値のビット列に特定のビットパターン（特異点）が含まれていれば，そのウィンドウの先頭位置をブロックの切れ目とする．特異点が含まれていなければウィンドウを 1 バイト移動して同じ処理を繰り返す．特異点は用いる重複排除ごとに固有のビット列が用いられる．ハッシュ値はランダム性があり，どのようなビット列を特異点に用いても，平均してどのビット列も一定頻度で出現するためである．可変長方式では次のようなパラメータが用いられる[5].

(1) 最小ファイルサイズ（基本は 40 バイト）
このサイズより小さいファイルは重複排除の対象としない．
(2) 最小ブロックサイズ（基本は 4,000 バイト）
ブロックの最小サイズ．
(3) 最大ブロックサイズ（基本は 16,000 バイト）
ブロックの最大サイズ．
(4) ウィンドウサイズ（基本は 32 バイト）
ハッシュ値に変換するブロック内の領域（スキャナー役）のサイズ．
(5) 特異点（基本は$(1234)_{16}$）
ウィンドウのハッシュ値に特異点が含まれていれば，そのウィンドウの位置を区切りとしてブロックを生成する．

ブロックの切れ目を検出するのに用いられるハッシュ関数は，ブロックのフィンガープリントを算出しているものとは異なり，ハッシュ値のランダム性より計算速度を重視したものが用いられる．これは，ブロックの切れ目を決めるものであって，データの重複を確認するために用いるハッシュ関数のようにランダム性を重要視しなくとも問題ないためである．このハッシュ関数には主にローリングハッシュ関数が用いられ，ラビン・カープ法というビット列検索方法と併用して用いられることが多い．

2.2 2 段階重複排除

2 段階重複排除（多段階重複排除）とは，従来の重複排除技術を段階的に行う方法である（図 2）．これは従来技術とは異なり，重複排除を段階的に行うことで，それぞれの段階で必要とされるリソースや処理時間を軽減することができる．加えて，バックアップサーバに保存される最終的なデータ量は従来技術とほぼ変わらないデータ削減率を維持することができるメリットがある．実際に企業などで重複排除を用いる際は，リソースなどに様々な制約条件が設けられるため，各段階の重複排除による負荷を調整できることは重要な要素のひとつである．

図 2　2 段階重複排除

3. 2 段階重複排除の分析

本研究は 2 段階重複排除のデータ削減の効果と，重複排除のための処理時間の特性を実環境にて明らかにし，分析するとともに，2 段階重複排除の組み合わせの違いによる特徴を捉える．2 段階方式に適した組み合わせを用いることで，データ削減率や処理時間の向上につながると考えられる．

4. 検証実験

4.1 実験方法

ファイルサーバを対象にして 2 段階重複排除と従来の 1 段階の重複排除を行い，処理時間やデータ削減量など測定して，2 段階重複排除の効果を分析する．実験対象は，大学の研究室内のファイルサーバである．このファイルサーバは，研究室に所属している学生がバックアップやデータの共有を意図して使用している．過去に約 60 人が利用したもので，ファイル数は 36,123 個，ファイルサイズは約 7 ギガバイトである．ファイルの種類には，ppt や word 等の文章ファイル，c や java 等のプログラムソースファイル，exe 等の実行形式，画像ファイルが多数含まれている．

2 段階方式と従来方式の特徴を対象とするファイルサイズごとに捉えるため，2 段階方式と従来方式のそれぞれについて，対象データ群のサイズを 1 ギガバイトから 1 ギガバイトずつ加算して，全体の 7 ギガバイトまで繰り返し実行する．各方式の組み合わせの構成を表 1 に示す．

表 1　各組み合わせの構成

組み合わせ	1回目（ブロックサイズ）	2回目（ブロックサイズ）
可変長・可変長	可変長ブロック方式（64KB～70KB）	可変長ブロック方式（4KB～16KB）
固定長64KB・可変長	固定長ブロック方式（64KB）	
固定長32KB・可変長	固定長ブロック方式（32KB）	
ファイル単位・可変長	ファイル単位	
1段階（可変長）	可変長ブロック方式（4KB～16KB）	

4.2 実験結果

ファイルサーバを対象にそれぞれの重複排除を実行した結果を図 3～6 に示す．結果を見ると，固定長ブロック方式と可変長ブロック方式の 2 段階方式に効率向上が見られた．処理時間はおよそ 20%から 35%の削減効果があり，生成ブロック数もおよそ 10%から 27%近くの削減効果が見られた．このデータ削減量と処理時間の両方の効果をひとつにまとめて見るために，

（削減効率）＝（データ削減量）／（処理時間）　　(1)

なる指標を導入すると，削減効率は従来方式に比べておよそ 23%向上している．可変長ブロック方式を 2 回行う 2 段階方式は，生成ブロック数の削減効果が見られたが，処理時間の大幅に増加したことにより，削減効率は従来の“1 段階”を下回る結果となった．なお，メタデータ量と生成ブロック数の増減傾向は似ている部分が多く見られたことから，メタデータ量は削減率のみを記載する．

効率向上が見られた“固定長 64KB・可変長”の 2 段階方式に着目すると，図 7 から 1 回目・2 回目のデータ削減量に大きな偏りが生じているのが分かる．2 回目の可変長はほとんどデータ削減を行わず，1 回目の固定長によって大幅なデータ削減が行われている．この事実を踏まえて，2 段階方式の組み合わせを変更して行った結果を図 8～10 に示す．削減効率のみに着目とした場合，“固定長 32KB・可変長”が最も高い数値を示し，約 47%の改善効果が得られた．加えて，削減データ量の低下もわずかである．ブロック数は“固定長 64KB・可変長”が最も少ない結果となった．削減効果の見られた各方式の結果を表 2 に示す．

表 2　各 2 段階方式の削減効果

組み合わせ	削減効率	データ削減量[MB]	処理時間[分]	ブロック数
1段階（可変長）	5.7	2,374	415	732,749
固定長64KB・可変長	7.0	2,185	310	620,401
固定長32KB・可変長	8.4	2,167	257	643,414
ファイル単位・可変長	6.9	2,374	344	659,172

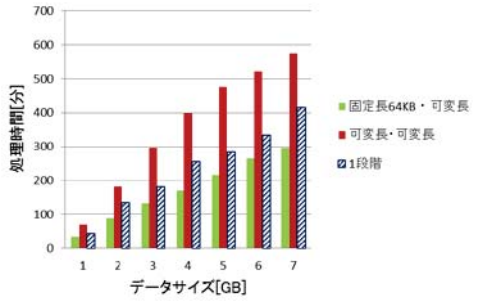

図 3　処理時間

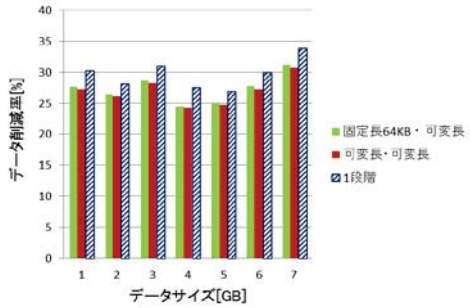

図 4　データ削減率

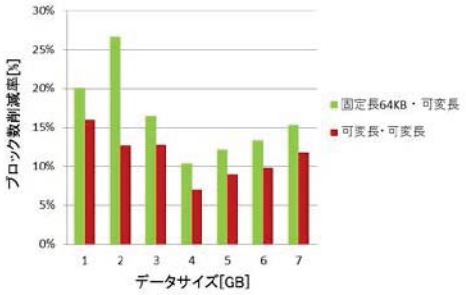

図 5　ブロック数削減率

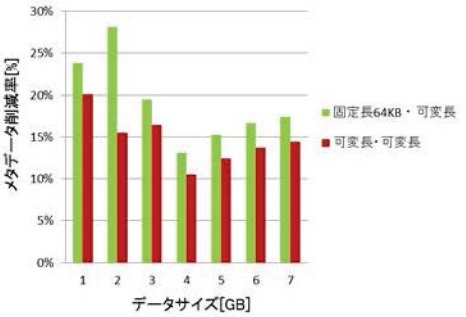

図 6　メタデータ削減率

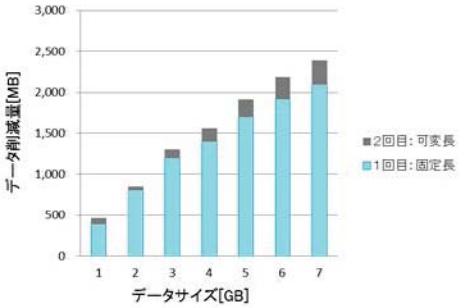

図 7　データ削減量割合（2 段階方式）

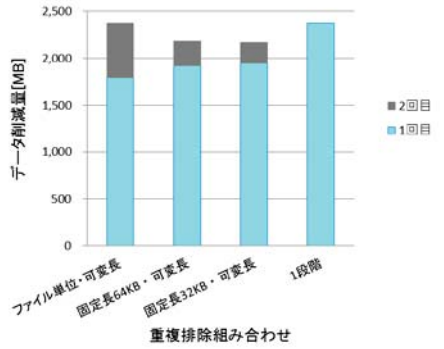

図 8　各データ削減量割合

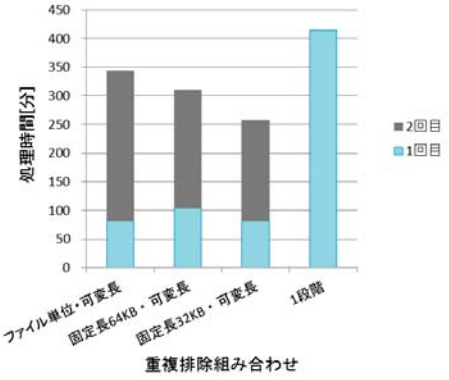

図 9　処理時間割合

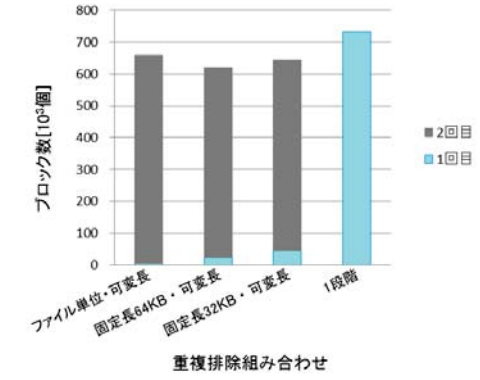

図 10　生成ブロック数割合

4.3 考察

本実験で効率向上が見られた固定長方式・可変長方式の2段階方式（“固定長 64KB・可変長”）の，重複排除 1 回目と 2 回目それぞれのデータ削減量を示した図 7 に示す．1 回目で行う固定長で削減されるデータ量は大きく,2 回目の可変長ブロック方式のデータ削減量は 1 回目に比べ，わずかな値であることが分かる．この結果から，ファイルサーバにはファイル単位で重複しているデータの割合が多く,ブロック単位の重複データが比較的少ないと考えられる．ファイル単位の重複排除を 7 ギガバイトのデータセットに対して実行した結果，データ削減量はおよそ 1.8 ギガバイトであり,ブロックサイズ 64 キロバイトの固定長方式とのデータ削減量の差はおよそ 0.1 ギガバイトであった．したがって,ブロック単位で検出できているデータは多くない．しかし，ファイル単位より処理量の多いはずの固定長方式を含む 2 段階方式の方が格段に少ない処理時間で済む結果となった．これは効率向上を図るうえで重要なのは，1 回目のデータ削減量を多くすることで，他に比べて処理が複雑な可変長方式の 2 回目に割り振るデータ量をいかに少なくできるかに左右される．“固定長 32KB・可変長”の結果からも，固定長方式のブロックサイズが小さくなったことで処理量が増えているのに対して，全体の処理時間が短くなったのは，1 回目のデータ削減量が向上して 2 回目に割り振るデータ量が減少したことに起因している．

表 2 の削減効率に着目した場合，ブロック数は比較的多いものの“固定長 32KB・可変長”が最も高い削減効率となった．これは，固定長方式で生成する 1 ブロックに要する処理時間が短いこと意味している．したがって，より小さいブロックサイズを用いて，固定長方式で得られる最大限の削減量に導くことが処理時間の短縮につながることを示している．

本実験の中で，従来技術の“1 段階”と同じデータ削減量を示したのは“ファイル単位・可変長”のみであった．これは他の 2 段階方式が，1 回目でファイルをブロック単位に切り分けるのに対して，“ファイル単位・可変長”は 1 回目をファイル単位で重複排除することによって,2 回目の可変長方式にファイル単位のままのデータを割り振れるためである．1 回目の重複排除で，固定長方式や可変長方式を用いてブロック単位にデータを切り出すことによって,本来であれば 2 回目の可変長方式によって検出可能な重複データも検出できなくなることがある．そのため，1 回目に固定長方式もしくは可変長方式を含む 2 段階方式には従来技術に比べてデータ削減量の低下が生じる．

図 5, 図 6 からメタデータ量は生成ブロック数に強く依存することが分かる．これはメタデータに含まれるデータのほとんどがブロックデータのハッシュ値であることに起因し，ブロック数の増減によりメタデータに書き込まれるデータ量も増減するためである．

5. 各組み合わせの特性

可変長ブロック方式を 2 回行う“可変長・可変長”は，生成ブロック数の削減効果を得ることができた．しかし，処理時間の大幅な増加やデータ削減量の減少が生じた．処理時間の増加によって削減効率は従来技術の“1 段階”より低下する結果となった．可変長方式を 2 回行うことは処理時間への影響も大きく，データ削減量の増加も見られなかったため，有効な組み合わせではない．

固定長方式と可変長方式を組み合わせた“固定長 64KB・可変長”は，生成ブロック数の減少に加えて処理時間の短縮効果が見られた．しかし，従来技術に比べデータ削減量が 3%前後低下する影響もあった．削減効率に着目した場合は，生成ブロック数および処理時間ともに削減効果があったことから，削減効率は向上する結果となった．2 段階方式の影響としてデータ削減量の低下が生じたが，削減効率は向上しており，有効な組み合わせである．

この組み合わせは，固定長方式を 1 回目に行うことによって，検出が容易な重複データを少ない処理量で先に検出する.そして,比較的処理量の多い可変長方式に割り振る，2 回目のデータ量を削減することで，効率化が図られている．従来技術に比べてデータ削減量が低下するのは，2 回目の可変長方式に割り振るデータがファイル単位のデータではなく，1 回目に生成されたブロック単位になっていることによる影響であると考えられる．

“固定長 32KB・可変長”は，“固定長 64KB・可変長”の固定長方式のブロックサイズを 32 キロバイトに変更した組み合わせである．生成ブロック数の削減およびに処理時間の短縮効果が見られた．“固定長 64KB・可変長”に比べ，生成ブロック数の削減効果は減少したが，処理時間の短縮効果はより良い結果が得られた．データ削減量は，3%程度の減少が影響として見られた．削減効率は組み合わせの中で最も高い値を示しており，“1 段階”に比べて約 47%の改善効果あった．これらの結果から，処理時間を優先するのであれば“固定長 32KB・可変長”が有効であり，データ削減量を優先するのであれば，生成ブロック数の削減によりメタデータ量の削減にもつながる“固定長 64KB・可変長”が有効である．

表 3　各組み合わせの特性

組み合わせ	メリット	デメリット
可変長・可変長	生成ブロック数　：　減少	処理時間　：　増加 データ削減量　：　低下
固定長64KB・可変長	生成ブロック数　：　減少 処理時間　：　減少	データ削減量　：　低下
固定長32KB・可変長	生成ブロック数　：　減少 処理時間　：　減少	データ削減量　：　低下
ファイル単位・可変長	生成ブロック数　：　減少 処理時間　：　減少 データ削減量　：　増減なし	特になし

固定長方式を含むこれら 2 つの組み合わせは，ブロックサイズによって異なる特性を示す結果となった．これは，1 回目の固定長方式による削減効果の違いから生まれており，ブロックサイズを 32 キロバイトに小さくしたことで，より細かい範囲で重複データを検出可能になり，生成ブロック数が増加した反面，2 回目に割り振るデータ量を削減できたことによる特性の違いである．

ファイル単位と可変長方式を組み合わせた“ファイル単位・可変長”は，生成ブロック数および処理時間の削減効果に加え，データ削減量の低下もなく，従来技術と同じデータ削減量を維持する結果となった．従来の“1 段階”に比べてデメリットとなる部分がなかった点が，この組み合わせのメリットといえる．ブロック数や処理時間の削減効果は固定長を含む組み合わせに比べ，効果が小さいため，データ削減量を優先する場合に限り有効な組み合わせである．

データ削減量の減少が見られなかった要因は，1 回目の重複排除がファイル単位であった点にある．他の組み合わせが 1 回目でブロック単位にデータを切り出しているのに対して，ファイル単位で重複を検出することで，2 回目の可変長方式に割り振るデータがブロック単位にならず，従来技術と同じ状態で可変長方式を行えるため，データ削減量の低下が起きない．

6. 結論

本研究で用いた 2 段階方式の各組み合わせに見られた，それぞれの特性を表 3 に示す．どの組み合わせも従来技術に比べ効率向上が見られ，それぞれ一長一短がある．その中でも特に，固定長方式を含む 2 段階方式は大きな効率向上が得られ，有効な組み合わせであることが分かった．実際に企業などで重複排除技術を導入する際には，様々な制約を設けることが常であり，処理時間やリソース，データ削減率など制約の範囲を守ることが最優先される．この組み合わせを用いた重複排除であれば，処理時間を優先する場合や，使用するリソースを優先する場合など，要望に合わせて組み合わせの設定値を変えることで，制約の範囲内で最大限の効率を得ることが可能である．

謝辞 本研究を行うにあたり，ご指導いただいた木下俊之教授ならびに小池到助手に感謝致します．また，日常の議論を通じて多くの知識や示唆を頂いた木下研究室の皆様に感謝します．

参考文献

1) C. Constantinescu, J. Glider, D. Chambliss: Mixing Deduplication and Compression on Active Data Sets, Data Compression Conference (DCC) 2011, pp.393-402, March 2011.
2) 尾形幹人，薦田憲久: 多段型重複排除による排除性能と排除率の改善，電気学会 情報システム研究会, IS-12-32, pp.47-51, 2012.
3) Open Source data deduplication for lessfs
http://www.lessfs.com/wordpress/
4) ZFS Deduplication
http://blogs.oracle.com/bonwick/en_US/entry/zfs_dedup
5) Dirk. Meister, Jürgen. Kaiser, Andre. Brinkmann, et al: A study on data deduplication in HPC storage Systems, SC 2012, IEEE Computer Society Press, Article 7.