Vol.2018-IFAT-130 No.7
Vol.2018-DC-108 No.7
2018/3/27

類似画像抽出を用いた重複排除の検証

吉田健太[†1]

概要：データ容量の削減として圧縮という手法があるが,計算コストや管理の面倒など問題点も多くある．そこで,ファイル単体として見る圧縮する従来手法に変わってストレージ全体を見て重複している部分を 排除する重複排除技術(Dedup:Deduplication) が開発された.Dedup はストレージ全体から重複している 情報を持っている部分を紐付けるため,使用しているデータ量が多いほど削除効率が多いが,同時にスト レージ全体を検索するため,コストが比例するという問題点を持っている．そこで,本研究では類似度に よる事前のグループわけ処理を行うことによる Dedup の効率化を提案する.なお,本研究では可変長ブ ロック方式を使用し,対象ファイルは一般的に使用頻度が高く使用サイズが大きいという点から画像ファ イルを使用する．類似度の抽出では画像の特徴を捉える SIFT を用い,実験の 400 個画像データ同士の 類似度を求めた,更に,類似度の高低の傾向が似ているものを閾値とハミング距離を元にグループわけを し,グループ毎に重複排除の実験を行った．

キーワード：重複排除，圧縮，画像解析

Verification of deduplication using similar image extraction

KENTA YOSHIDA[†1]

Abstract: there is a compression method to reduce data capac- ity, there are also another problems such as cost of calculation and troubles of management. Therefore, instead of the conventional method of compressing a single file, deduplication technology (Dedup: De-duplication) has been devel- oped to eliminate redundant parts by looking at the entire storage. As Dedup associating parts that have duplicate information from the entire storage, the deletion efficiency is higher as the amount of data used is higher, but at the same time, is proportional with the cost of searching the entire storage. There- fore, we propose a way to improve Dedup's efficiency by prior grouping process based on similarity. In this research, variable length block method is used and image file is used from the viewpoint that the target file is generally used frequently and the size of usage is large. In the extraction of similarity, SIFT is used to capture the features of the image and found the similarity between 400 image data used in the experiment. Then, we grouped items whose similarity trends are similar to each other based on the threshold value and Hamming distance, and conducted an experiment of deduplication for each group.

Keywords: Data Deduplication, Compress, Image analysis

1. はじめに

近年,計算処理の向上に伴い大規模なシステムが利用され ,バックアップシステムの普及により使用されるストレージも増大している．これらの巨大なデータを企業や個人のローカルストレージに保持をしておくにはコストがかかりすぎるため,クラウドコンピューティングというサービスを利用しする企業・個人も増えている.そういった中,従来の圧縮技術だけでは満足なデータ削除が望めず,多人数が使用する際に解凍のコストがかかり費用対効果が望めないケースが増えている．そこで,個々のファイルの情報を見て圧縮する従来の技術に代わりストレージ全体を検索して重複する部分のデータ削除を行う手法が近年研究されている．

重複排除システム Data De-duplication(以後 Dedup) はその一つである.Dedup はストレージの使用容量が多いほど削除効率が上がり,データ圧縮との併用が可能であるため高い関心を集めている．

2. Dedup と問題点

2.1 Dedup

データ重複排除とは,ファイ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する.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技術のことを指す．重複排除または Dedup とも呼ばれる．

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．ファイル重複判定では,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．

†1 東京工科大学
Tokyo University of Technology.

算出されたハッシュ値はFingerPrintとも呼ばれ.それぞれのブロックを一意に判別するため識別子として用いられる.
これらの識別子は,ユニークな識別子に対して参照される,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示し,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,その識別子はユニークな識別子として判断され,新しくデータベースに保存される.新しく認識に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.

2.2 ファイル単位方式

この方式ではあるファイル A と中身が全く同一のファイル B を重複と見つけ,1 つにまとめる方法である.似たもの単一インスタンス記憶域が WindowsOS に搭載されている.動作が非常に軽く効果を上げるが,近年の PC の発達に伴い,より多くの重複を見つけることが求められている.

2.3 固定長ブロック方式

ブロック単位ではブロックごとに重複の有無を走査している.ブロック単位の重複排除は固定長ブロック方式と可変長ブロック方式の 2 つである.
固定長ブロック方式ではファイルを一定のブロックサイズに分割し,ブロックごとに重複判定を行う.この方式を採用する重複排除機能を持ったソフトは,多くが 4KB から128KB の間でブロックを生成しており,一部のソフトではユーザーが重複排除を行う対象に合わせてブロックサイズを選択肢から選べるソフトも存在する.
ブロックサイズを選択できるようにしているわけは,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上昇するが,その反面 1 つのファイルにつき生成されるブロックの数が増加してしまい,データベースに格納される識別子にかかるレスポンス速度が低下する,その逆にブロックサイズを大きくすればファイル単体で生成されるブロックの数が減少し,重複確認にかかる時間は比較的短くなりレスポンス速度は上昇する.しかし,ブロックサイズをおおきくしたことでデータが重複する確率は格段に低下してしまい,重複排除率も低下してしまう.
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合わせてブロックサイズを変更する必要がある.

2.4 可変長ブロック方式

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を切り出すポイントを決めている,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,はじめにブロックデータの基盤となる一定量のデータ (オフセット) をデータストリームから取得する.加えて,一定量のデータ (ウィンドウ) を読み込み,これをハッシュ化した文字列から,特定の文字列パターン(特異点) が確認できたときにブロックの生成ポイントとする.
ブロックの切り出し位置の判別に用いられるハッシュ関数は,ブロックのフンガープリントを算出してるものとは異なり,ハッシュ値のランダム性を重視した関数ではなく,ランダム性より計算速度を重視したハッシュ関数が用いられる.これは,ブロックの切り出し位置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視せずとも問題ないためである.ブロックの切り出し位置決定に用いるハッシュ値を少ハッシュと呼ぶ.
このようにデータパターンに基づいてブロックを切り分けていくようにすれば,たとえファイルへのデータ挿入がズレを生じさせたとしても,ブロックを切り分けるタイミングはほとんど変わらない.ブロックの切り分け位置は先頭からの長さではなくデータパターンに基いてきまるためである.これにより,バックアップソフトウェアがいくらファイルを纏めなおしても,重複排除を引き続き効率的に行える.
この可変長ブロック方式という技術は,バックアップサーバ側に重複効率をよく行うための重複技術の一つになっており,近年では主流とされている.データ重複排除で大きく謳われているデータ量を 20 分の 1 にするメカニズムは可変長ブロック単位の重複排除を使うことを前提としている.

2.5 問題点

だが,問題点として Dedup はストレージを多人数で使用するなどして類似性の高いデータ群に対しては高い効果が得られるが,類似性の低いデータ群には効果があまり出ない.その点から使用前に対象ストレージの特性を確認する必要がある. また,ストレージ全体の検索をかけるため CPU やメモリなど計算資源に大きな負荷がかかる.この二点から,コストをかけたが期待していた削除効果を生み出せないという場合が存在する.
そのため,Dedup による重複排除の実行前に削除効果の予測と実行コストの削減が必要不可欠である.

Vol.2018-IFAT-130 No.7
Vol.2018-DC-108 No.7
2018/3/27

3. 提案手法と評価方法

本章では 3.1 節において提案手法を説明し,3.2 節において評価方法についてまとめる.
なお,一般的に使用頻度か高く使用サイズが大きいという点で本件研究では画像データを対象に行う.

3.1 提案手法

まず,対象のファイルを OpenCV に付属されている SIFT のデコーダーを使用して特徴点を算出する.各画像で抽出した特徴点同士を総当りで比較を行い,その際ユークリッド距離を用いる.ユークリッド距離が一番小さい点同士を対応点とする.その対応点のベクトルの距離が特徴点毎の類似度となる.求められた特徴点の距離を特徴点の数で割り平均値を画像の類似度とする.算出された画像の類似度を閾値でバイナリ化し,ハミング距離の平均値を取ることによって類似度の傾向を数値化することで傾向類似度が取得できる.それをもとに画像デー タのグループ分けを行う.グループの分け方はまずすべての画像データの類似度を総当りで任意の類似度でトリミングを行い”グループ A”と呼称する.残ったデータから再度ハミング距離を算出し,トリミングを行い”グループ B”とする.同じ作業を繰り返し,グループをわける.残った画像データはどれにも類似していないデータ群として”他”と呼称する. そしてそれらのグループごとにブロックリストを作成し,グループ内での重複排除を行うことによって従来の 1 つのブロックリストを使用する手法より高速な重複削除を実現する.

3.2 評価方法

評価方法として,重複排除量率と所要時間を用いる.重複排除率は,重複排除後データ量/元データ量*100 によって導き出すことができる.まず既存の Dedup を用いてファイル全体に重複排除を行う.その結果を元に提案手法との結果の差によって割合を導き出す.

4. 実験結果

重複排除は 3 つのパラメータが必要であり、特異点サイズを 14bit[1]ブロック最大長を 200Byte[2]最小ブロックサイズを 500Byte[2]設定した。
これらの数値は参考文献における最適値である。

4.1 類似度取得

本実験で使用する画像データはデータセットとして公開している COIL-20[3]60 枚 (obj[1-5] *.bmp) と自然や大学の校舎など画像データ (etc[1-40].bmp)40 枚,計 400 枚のデータを使用する.
obj2__00 とそれ以外の特徴量をグラフ化したしたものを下記の図 1 に表す.

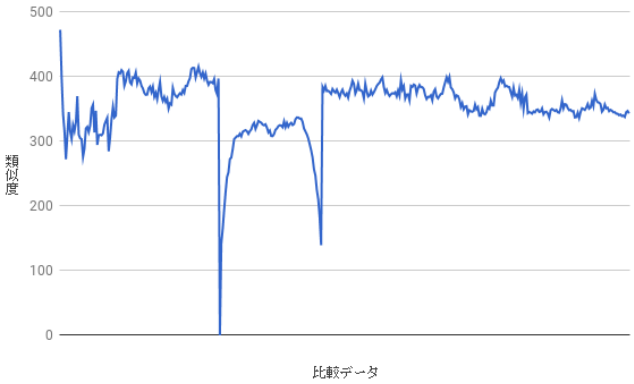

図 1 obj2__00 の類似度グラフ
Fig. 1 Similarity graph of obj2__00

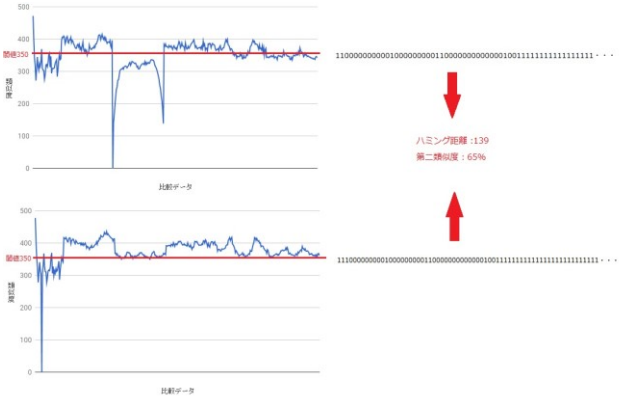

図 2 二つのデータからハミング距離取得
Fig. 2 Get Hamming distance from two data

4.2 グループ分け

グループ分けに使用する閾値は 350 とし,ハッシュ化しハミング距離を取る.(図 2)
傾向類似度から選抜したグループ分けの結果を下記の表に示す.今回の実験ではグループ分けする傾向類似度を 90%85%80%75%とし,グループを 5 つに設定した.

表 1 グループ選抜結果
Table 1 Group selection result

	ファイル数						
	グループA	グループB	グループC	グループD	グループE	その他	計
全て							400
90%以上	70	62	40	30	35	173	400
85%以上	102	88	46	44	32	88	400
80%以上	146	108	69	20	19	38	400
75%以上	205	121	34	23	9	8	400

4.3 重複排除

400 枚画像データで従来方式の重複排除とグループ分けしたデータブロックでの重複排除を比較した結果は表 2 に示す.

表 2　重複排除結果

Table 2　Deduplication result

	類似度抽出	重複排除サイズ(Byte)	重複排除率(%)	所要時間(秒)
既存方式		30,503,295	42.7	780.5
提案方式	90%以上	30,323,007	42.5	714.8
	85%以上	30,337,714	42.5	695.6
	80%以上	30,336,069	42.5	697.0
	75%以上	30,336,640	42.5	692.0

表 3 グループ毎の重複排除結果

Table 3 Deduplication result for each group

類似度抽出	重複排除対象	ファイル数	重複排除サイズ(Byte)	所要時間(秒)
90%以上	グループAのみ	70	5812677	128.1
	AとB	132	8799192	218.0
	ABC	172	9938162	255.1
	ABCD	202	11945386	310.0
	ABCDE	227	14228891	357.0
	ABCDEその他	400	30323007	714.8
85%以上	グループAのみ	102	8839572	190.1
	AとB	190	13960354	329.7
	ABC	236	15390518	373.9
	ABCD	270	18398259	455.3
	ABCDE	321	22254290	526.9
	ABCDEその他	400	30337714	695.6
80%以上	グループAのみ	146	13274694	286.5
	AとB	254	19538338	456.1
	ABC	323	22927332	540.3
	ABCD	343	24316514	578.3
	ABCDE	362	26723344	623.5
	ABCDEその他	400	30336069	697.0
75%以上	グループAのみ	205	15288964	367.7
	AとB	326	24436673	554.5
	ABC	360	26798980	615.7
	ABCD	383	28407550	656.9
	ABCDE	329	29651030	677.8
	ABCDEその他	400	30366640	629.0

グループ毎の詳細な結果は表 3 で示す.グループ A のみで重複排除を行った結果,グループ A と B で重複排除行った結果といった様にグルー プを重ねていき,最後はその他を含めた 400 枚の画像ファイルデータで重複排除を行った.

5. 評価

全体として見ると,表 2 から重複排除率はほぼ変わっていないが,所要時間を最小 65.2 秒最大 88.5 秒短縮に成功した.割合で示すと,最小 8.3%最大 11.3%の高速化が行えた.
グループわけに関しては表 3 からグループ選抜の際の類似度が高いほど,起点のグループでは所要時間が短いものの,グループ数が増えていくとかなり増大してしまう.反対にグループ選抜の際の傾向類似度が低いほど,起点のグループでは所要時間が長いものの,グループ数による影響が少ない結果となった.
以上のことを踏まえ,各傾向類似度ごとの結果についてはグラフを作成し,以下の図で示す.以下のグラフから傾向類似度 90%85%80%でグループ分けした場合グループ A と B 以降は重複排除のサイズが低迷することがわかった.このことからグループ A と B のみグループ毎に重複排除を行い,C と D と E とその他はまとめて重複排除を行ったほうが費用対効果として良いと言える.

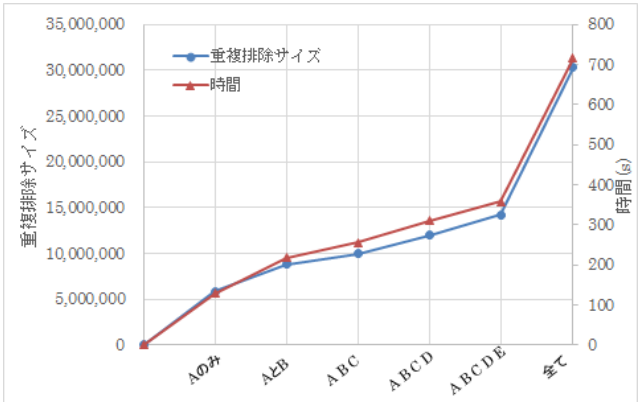

図 3　傾向類似度 90%によるグループ分けの重複排除サイズと時間

Fig. 3 Deduplication size and time for grouping by trend similarity 90%

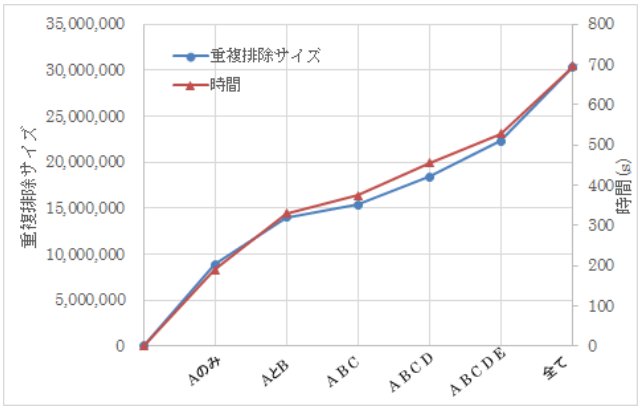

図 4 傾向類似度 85%によるグループ分けの重複排除サイズと時間

Fig. 4 Deduplication size and time for grouping by trend similarity 85%

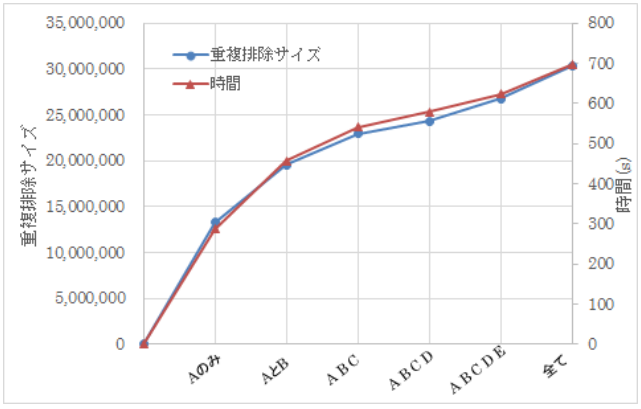

図 5 傾向類似度 80%によるグループ分けの重複排除サイズと時間

Fig. 5 Deduplication size and time for grouping by trend similarity 80%

Vol.2018-IFAT-130 No.7
Vol.2018-DC-108 No.7
2018/3/27

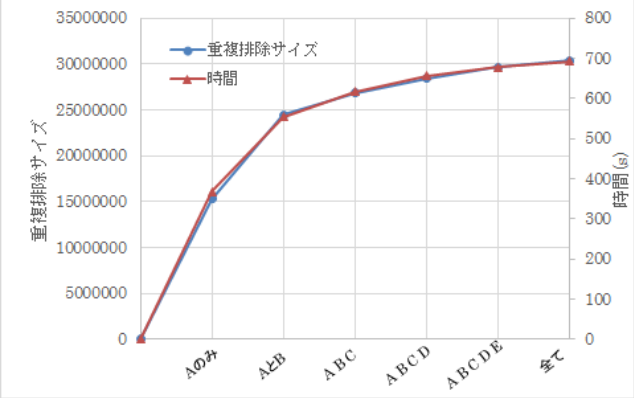

図 6 傾向類似度 75%によるグループ分けの重複排除サイズと時間

Fig. 6Deduplication size and time for grouping by trend similarity 75%

6. 結論

近年データ量が増えていくことでストレージにたいする管理コストも増えていく.しかし増加するデータ量に対して新たに圧縮技術を導入するケースは少なく,大容量ストレージを購入し追加するケースが増えている.理由の一部として既にある大量のデータに対して,圧縮・解凍を行うことは計算資源の消費量が増大になってしまう.重複排除にも同一のことが言え,圧縮率と処理負荷はおおよそ比例してしまい,ケースバイケースとなってしまうことが多い.

そこで本研究では一般的に使われている画像データのみだが,既存の重複排除を用いて重複排除率をなるだけ落とさず,所要時間を 10%以上減らすことに成功した.

さらに本研究では要所要所のアルゴリズムを変更し,改良しやすいシステムになっているので応用が効きやすいものとなっている.

謝辞 本研究にあたり,終始適切な助言を賜り,また丁寧に指導して下さった木下 俊之先生及び小池 到氏にこの場を借りて感謝の意を表します.

参考文献

[1] Matrazali noorafiza, itaru koike, hiroto yamasaki, abdulhashim rizalhasrin, toshiyuki kinoshita , “lock length optimization in data deduplication technique.”,” Data Compression Conference, 2009. DCC '09.”

[2] 荻原 美絵, 高谷 美月, 粕谷 輝久, 小池 到, 木下 俊之 “重複排除における特異点サイズの最適化”, 情報処理学会研究報告, vol.2014-mps-99 no.2.

[3] Software: Coil-20: Columbia object image library: http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.