Vol.2014-MPS-99 No.2
2014/7/21

重複排除における特異点サイズの最適化

荻原 美絵[†1]　高谷 美月[†1]　粕谷 輝久[†1]
小池 到[†1]　木下 俊之[†1]

近年，企業の計算機システム内には大量のデータが蓄積され，その中には複数のバージョンのファイルを保持することも多く，全く同一または殆ど同一の重複したデータが大量に存在している．重複排除は，これら同一のデータのうちの一つのみを保存し他を削除することで，データ量を大幅に削減するデータ圧縮技術の一つである．
本研究は，重複排除技術のうち可変長ブロック方式について，ブロックを生成する際に用いる特異点のサイズが重複排除の効果にどう影響するかを評価した．重複排除の効果は，特異点サイズや生成されるブロック数に影響される．特異点サイズを 4 ビットから 23 ビットに変化させて重複排除を行い，データの削減量を示す重複排除率の変化を調べた．その結果，最適な特異点サイズは 15 ビットであり，特異点サイズがこれより大きいかまたは小さい場合に比べて，重複排除率が最大で約 7%向上することを確認した．

Singularity Size Optimization in Data Deduplication

Mie Ogiwara[†1]　Mizuki Takaya[†1]　Teruhisa Kasuya[†1]
Itaru Koike[†1]　Toshiyuki Kinoshita[†1]

Recently, massive data growth and data duplication in enterprise systems have led to the use of deduplication techniques. Since we keep multiple versions of files, there may be a large volume of mostly or exactly identical data. Deduplication is a powerful storage optimization technique that can be adopted to manage maintenance issues in data growth. We evaluated the effect of deduplication by analyzing how the singularity size affects the effect of deduplication for variable-length blocks. We clarify that the effect of deduplication is affected by the singularity size and the number of created blocks. We traced the change in the deduplication rate, which indicates the reduce ratio of file data volume, by changing the singularity size from 4 bits to 23 bits. The result shows that the optimum singularity size is 15 bits and that the deduplication rate is improved around 7 % at the optimum singularity size compared with at smaller or larger singularity size.

1. はじめに

近年，音楽や映像などを含むマルチメディアデータや大容量のデータベースの利用が増加し，企業内で蓄積されるファイルデータが大幅に増加している．これらの企業データは複数バージョンの管理がされている場合が多く，全く同一かまたは殆ど同一のデータが多く存在していて，これらに重複排除技術を用いることにより，データ量を大幅に削減することができる．

図 1.1 に示すように，重複排除を行う際には複数のレコードの類似性を計算し，類似あるいは同一が検出された場合はそのうちの一つを代表データとして残し，他はこの代表データを指し示すリンクに置き換えられる．このようにファイルシステム内の重複データ（冗長データ）を排除することで，データ量を大幅に削減することができる．この結果，データストレージの効率を大幅に向上させることができ，データのメンテナンスやストレージコストも減少すると考えられる．

重複排除には，ファイル単位に行う方法とブロック単位に行う方法の 2 種類がある（図 1.2（a）（b）参照）．ファイル単位の重複排除は，ファイル全体が同一の場合にそのうちのひとつを残し他方を削除する．一方，ブロック単位の重複排除では，ファイルをブロックと呼ばれるいくつかの部分に分割し，重複するブロックごとに削除する．ファイル単位の重複排除はファイルをブロックに分割する必要がないので実行時間が短くてすむが，ファイル全体が同一でない限り重複排除できないので，重複排除の効率を上げることが難しいという欠点がある．一方，ブロック単位の重複排除はファイルが厳密に同一である必要はないのでより効率的に重複排除を実行できるが，ファイルを重複排除に適したブロックに分割する必要があるため，実行時間が長くなるという欠点がある．

ブロック単位の重複排除には，2 つの方法がある．一つはブロックサイズが一定な固定長ブロック方式であり，他方はブロックサイズが変化する可変長ブロック方式である．可変長ブロック方式では，できる限り同一のブロックが多くなるようにブロックの境界を決める必要がある．このために，ブロックの境界の候補ごとに決められたハッシュ関数を用いてハッシュ値を算出し，このハッシュ値に特異点と呼ばれる特定のビットパターンが含まれていればブロックの境界とし（従ってここでブロックが生成される），特異

†1 東京工科大学コンピュータサイエンス学部
School of Computer Science, Tokyo University of Technology

点が含まれていなければブロックの境界の候補から外す．これはハッシュ値に特異点が含まれているもの同士ならブロック全体が同一である確率が高いと考えられるからである．このハッシュ値と特異点を用いてブロックを決める方法では，重複排除の効果は特異点の影響を大きく受ける．特異点がハッシュ値に含まれるか否かは，確率的に特異点のビットパターンの内容よりも特異点のサイズ(ビット数)により強く影響を受けると考えられる．本研究の目的は，重複排除の効果を最大にするような最適な特異点サイズを解析することにある．

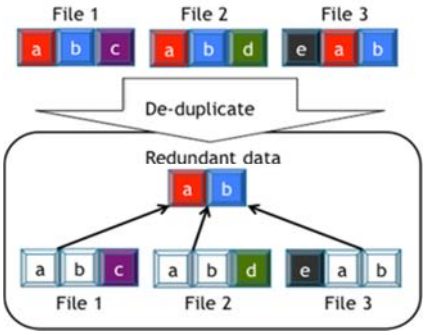
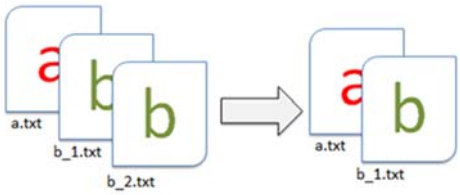
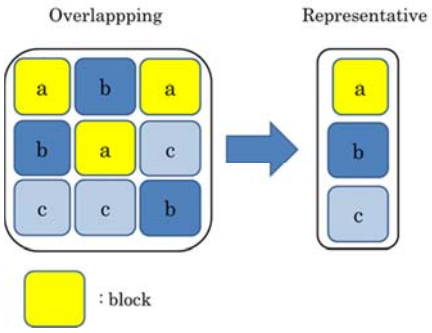


図 1.1 重複排除の原理



(a)ファイル単位の重複排除



(b)ブロック単位の重複排除

図 1.2 2 種類の重複排除

2. 関連研究

可変長ブロック方式ではブロック長が変化するが，極端に大きかったり極端に小さいブロックが生成されないように，最大ブロック長と最小ブロック長が設定される．重複排除の効果は，この最大／最小ブロック長に影響を受ける．

[1] では，ブロック長が 4～16 K バイトの固定長ブロック方式の重複排除の効果を調べた．[3]では，可変長ブロック方式の効率が議論された．さらに[2]では，ブロック長が 4 K バイトよりも大きい場合について，可変長ブロック方式と固定長ブロック方式の重複排除の効果の違いが報告され，[4]ではブロック長が 4 K バイトより小さい場合について同様の報告がされた．これら[1]～[4]はいずれもブロック長と重複排除の効果との関係を調べているが，特異点サイズを扱ったものではない．本研究では，重複排除の効果を最大にするような最適な特異点サイズを解析する．

3. ブロック単位の重複排除

3.1 ブロックの 2 つの単位

ブロック単位の重複排除には固定長ブロックと可変長ブロックの 2 種類がある．図 3.1～3.3 により，これら 2 種類

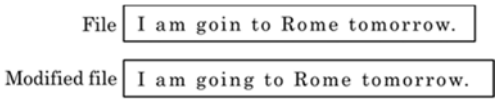


図 3.1 例文

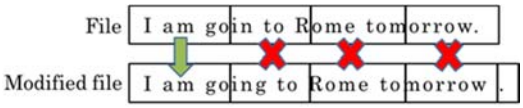


図 3.2 固定長ブロック(文字数 7 固定)

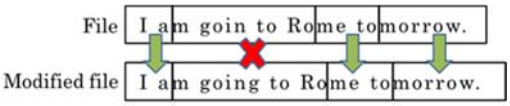


図 3.3 可変長ブロック(‘m’で始まるブロック)

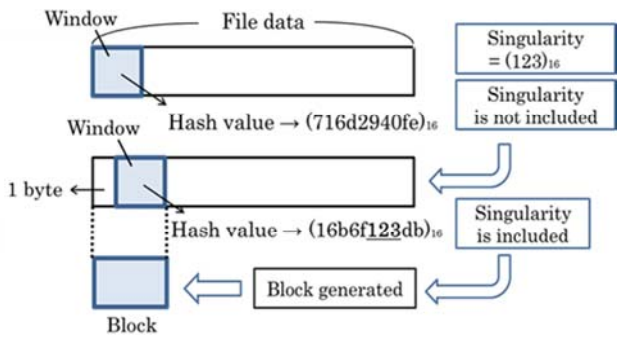


図 3.4 可変長ブロック生成メカニズム

の方法について詳しく説明する．図 3.1 に示すように“I am goin to Rome tomorrow.”という文を考えると，この文には誤りがあり，修正のために“g”を挿入すると“I am going to Rome tomorrow.”という文になる．固定長ブロック方式では，ブロック長が一定のため文字の挿入があるとデータがずれて，挿入位置より後ろのブロックは元とは異なるブロックと判断され重複排除できなくなる．一方，可変長ブロックは様々な長さでブロックを区切ることができるので，中央付近に挿入があっても不一致になるのはそのブロックだけで前後のブロックは変わらない．このため，挿入位置より後ろのブロックについても重複排除が可能になる．このように可変長ブロック方式は，データの挿入・削除があっても重複排除の効果を維持することができる．

3.2 可変長ブロック

可変長ブロックの生成には，Rabin-Karp algorithm を用いる．このアルゴリズムには，次のようなパラメータが用いられる．

(1) 最小ファイルサイズ（基本は 40 バイト）
このサイズよりも小さいファイルは，重複排除の対象とされない．
(2) 最小ブロック長（基本は 4,000 バイト）
(3) 最大ブロック長（基本は 16,000 バイト）
(4) ウィンドウサイズ（基本は 32 バイト）
ウィンドウは，ハッシュ値を算出する単位である．
(5) 特異点（基本は 123 ）
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．

最初に最小ファイルサイズより大きいファイルが読み込まれる．これが最小ブロックサイズより小さければ，そのままこのファイル全体が 1 つのブロックとして生成される．最小ブロックサイズより大きければ，ブロックの区切り点探索を行う．区切り点探索は，まず最小ブロックサイズの位置のウィンドウについてハッシュ値を求め，生成されたハッシュ値が特異点を含めばそこを区切り点としてブロックを生成し，特異点を含まなければウィンドウを 1 バイトずらして再び区切り点探索を繰り返す．最大ブロックサイズまで区切り点が見つからなければ，そのまま最大ブロックサイズのブロックを生成する．

4. 最適な特異点サイズの検証

4.1 検証実験

重複排除の効果の指標として「重複排除率＝重複排除後のファイルサイズ／元のファイルサイズ」を用いる．重複排除率が大きいほど，重複排除の効果が高いことを示している．検証のための実験は可変長ブロック方式を用いて，東京工科大学の我々の研究室のファイルシステムで行った．ファイルシステムの総容量は約 7G バイト，ターゲットファイルはテキストデータ（doc，xls，ppt 等），画像データ（pdf，gif，bmp，jpg）などを含んでいる．最大／最小ブロック長は，通常の重複排除で用いられる 4K バイトと 16K バイトに設定した [1].

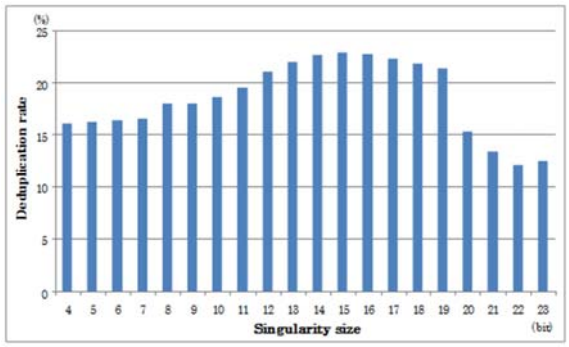


図 4.1　特異点サイズと重複排除率

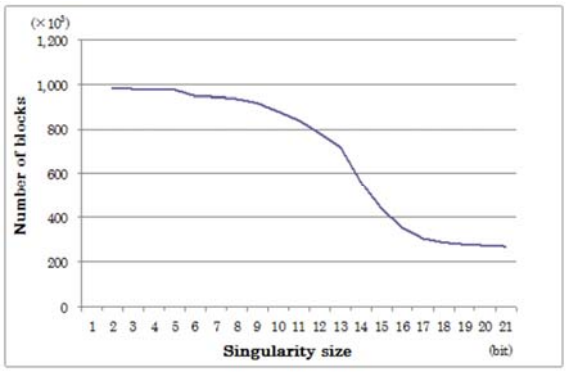


図 4.2　特異点サイズとブロック数

4.2 実験結果

図 4.1 と図 4.2 は，特異点サイズと重複排除率，ブロック数との関係をそれぞれ示している．図 4.1 によると，特異点が 15 ビットのときに最も高い重複排除率を示している． また図 4.2 によると，特異点が増加するにつれて，ブロック数は減少している．これは特異点サイズが大きくなると，ウィンドウのハッシュ値に特異点が含まれにくくなるためと考えられる．

特異点サイズが 15 ビットより小さいときは，特異点サイズが大きくなると重複排除率が高くなっている．これはこの範囲で特異点サイズが大きくなると，重複排除に適したブロックがより多く生成されるためと考えられる．特異点サイズが小さすぎると，ほぼ全てのウィンドウのハッシュ値に特異点が含まれて多数の最小ブロック長のブロックが生成されてしまう．図 4.3 と表 4.1 によると，特異点サイ

ズが 7 ビットと 11 ビットのときサイズが 4～5 K バイトのブロックは全ブロックの 98%および 97%を占めている. これは事実上，固定長ブロック方式とほぼ同じであり，可変長ブロック方式の利点が生かされていないことが分かる.

一方, 特異点サイズが 15 ビットのときに，重複排除率は最も高くなっている．このとき，最小ブロックは全ブロックの 63%を占めるが，これは特異点サイズが 7 ビット，11 ビットのときの 98%，97%に比べてかなり小さい割合である．重複排除に適したサイズのブロックが，より多く生成されているためと考えられる．このように特異点サイズを最適化することにより，より効果的な重複排除を行うことができる.

特異点サイズが 15 ビットより大きくなると, ウィンドウのハッシュ値に特異点が含まれにくくなり，より長いブロックが生成されているためと考えられる. 図 4.2 によると，特異点サイズが大きくなるにつれてブロック数は減少し，表 4.1 によると最大ブロック長のブロック数は急激に増加していることが分かる. 特異点サイズが 15 ビットのとき最大ブロック長である 15～16 K バイトのブロックは全ブロックのわずか 0.6%であるが，特異点サイズが 19 ビット，23 ビットになると 15～16 K バイトのブロックは全ブロックのそれぞれ 48%，90%を占めている．この場合も可変長ブロック方式の効果は薄れ，固定長ブロック方式に近くなっていると考えられる.

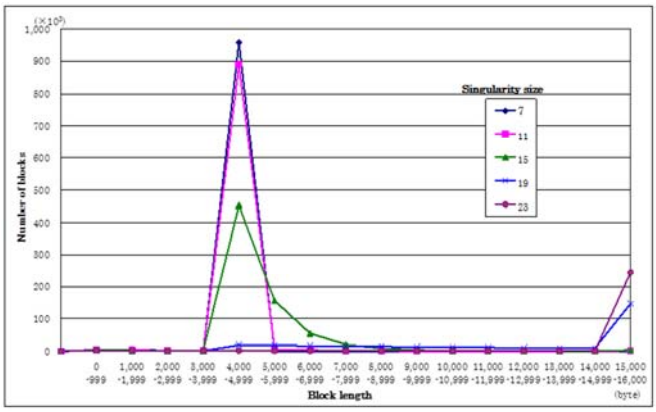


図 4.3 ブロック数とブロック長

表 4.1 ブロック数と特異点サイズ

Range of block size \ Singularity size	7 (bits)	11	15	19	23
0～ 4,000 (byte)	14,829 (1.5%)	14,132 (1.5%)	12,306 (1.7%)	8,651 (2.8%)	7,972 (2.9%)
4,001～ 5,000	957,374 (98.4%)	888,619 (97.1%)	452,886 (63.1%)	19,077 (6.2%)	2,110 (0.8%)
5,001～15,000	1,335 (0.1%)	11,620 (1.3%)	248,069 (34.6%)	131,201 (43.0%)	15,744 (5.8%)
15,001～16,000	43 (0.0%)	1,079 (0.1%)	4,022 (0.6%)	146,528 (48.0%)	244,647 (90.5%)
Total	973,581	915,450	717,283	305,457	270,473

5. 終わりに

本研究では，最適な特異点サイズを用いることにより，重複排除の効果を高められることを明らかにした．最適な特異点サイズは 15 ビットで, 最適でない場合に比べて重複排除率が最大で約 7%向上していることが確認された.

今後の課題として，最大／最小ブロック長を 4／16 K バイトに固定せずに他の様々なブロック長について最適な特異点サイズを解析することが必要である.

参考文献

[1] Q. He, Z. Li, X. Zhang, “Data deduplication techniques,” Future Information Technology and Management Engineering (FITME) 2010, vol. 1, pp. 430-433, Oct. 2010

[2] C. Constantinescu, J. Glider, D. Chambliss, “Mixing Deduplication and Compression on Active Data Sets,” Data Compression Conference (DCC) 2011, pp. 393- 402, March 2011

[3] A. N. Yasa, P. C. Nagesh, “Space savings and design considerations in variable length deduplication,” ACM SIGOPS Operating Systems Review, Vol. 46 Issue 3, pp. 57-64, Dec. 2012

[4] M. Noorafiza, I. Koike, H. Yamasaki, A. Rizalhasrin, T. Kinoshita, “Block Length Optimization in Data Deduplication Technique,” Proceedings of the 10th International Conference on Scientific Computing (CSC2013), pp.216-220, July 2013