Vol.2012-DPS-150 No.39
Vol.2012-CSEC-56 No.39
2012/3/1

大規模仮想ディスクにおける重複排除

遠藤 立[†], 熊耳 順一[†], 上原 稔[†]

低コストの大容量ストレージに対する要求は非常に高い。我々は、空容量を集約してこのようなストレージを構築するために、ディスクレベル分散型ストレージを構築するためのツールキット VLSD (Virtual Large Scale Disks)を開発した。VLSD は空き容量を有効活用することで低コストの大規模ストレージを実現する。一般に、ストレージのコストはディスク台数に比例する。ディスク台数を削減する方法に重複排除がある。近年、ストレージにおける重複排除が注目されている。我々は、VLSD を用いてインライン方式に基づく重複排除を行う仮想ディスク DedupFolderDisk を実装した。この仮想ディスクでは、同一内容のブロックを1つのフォルダに集約する。本論文では DedupFolderDisk の設計と実装について詳細に述べ、その性能および容量効率を評価する。DedupFolderDisk は中規模のストレージに適するが、重複排除の効果は高い。特に、バックアップなど複数ユーザが同じデータを所持する場合に効果的である。一方、性能は低い。性能とブロックサイズはトレードオフの関係にある。

Deduplication in Virtual Large-Scale Disks

Ritu Endo[†], Junichi Kumamimi[†], Minoru Uehara[†]

Recently, the demand of low cost large scale storages increases. We developed VLSD (Virtual Large Scale Disks) toolkit for constructing virtual disk based distributed storages, which aggregate free spaces of individual disks. VLSD realizes low-cost large-scale storage by collecting free disk spaces from many computers. Generally, the cost of storage is proportional to the number of disks in the storage. Deduplication is effective for reducing the number of disks. Recently, deduplication is one of hot topics in storage. We have developed DedupFolderDisk, which is a virtual disk that deduplicates blocks on basis of inline method. In this virtual disk, the same contents are stored into a folder. In this paper, we describe the design and implementation of DedupFolderDisk and evaluate its performance and capacity efficiency. From the result of these evaluations, we conclude that DedupFolderDisk is suited for middle-scale storage and it is effective in a sense of deduplication. Especially, it is useful when users has same contents. In the other word, its performance is not so good. Therefore, the relationship between performance and block size is trade-off.

[†]東洋大学 工学部 情報工学科
Faculty of Information and Computer Sciences, Toyo University.

1. はじめに

ストレージ技術の進歩とクラウドの普及により、オンラインストレージに対する要求はますます高まっている。クラウドのような大規模ストレージでは、大量のディスクを使用する。ストレージのコストはディスクの台数に比例する。よって、ストレージのコストを下げるには、ディスク台数を減らすことが重要である。ストレージの容量を一定に維持したままディスク台数を減らすには、ストレージの容量効率を高めるか、1 台のディスクに格納可能な情報量を増やせばよい。

前者は RAID に対して有効である。多くの大規模ストレージは RAID[1)2)]で構成されている。我々は高信頼 RAID を構成する技法として階層型 RAID[3)]、N 進 RAID[5)6)7)12)]、直交 RAID[11)14)15)]等について検討してきた。RAID に関しては、理論的に、m パリティを用いて m 耐故障 RAID が構成可能であることが知られている 16)。よって、容量効率の上限が存在する。

一方、後者として、データ圧縮、重複排除などの方式がある。データ圧縮方式では、LZ77(zip)、LZMA(7-zip)などの符号化法によりデータを圧縮することで論理的な情報量を拡大する。文献 17)によれば、様々な圧縮方式による圧縮率の平均は約 20%である。一方、重複排除方式では、同じ（重複する）データを排除することで資源を有効活用する。重複排除には、それを行うデータ単位およびタイミングによる様々な方式がある。詳細は関連研究にて紹介する。文献 18)によると重複排除の効果は 20:1 すなわち 5%も十分現実的な値である。これらの結果を比較すると、データ圧縮より重複排除の方が物理資源を削減する効果が高いように見える。しかし、この前提には、重複排除がバックアップなど限られた状況で利用されていることを見逃すわけにはいかない。それでも、重複排除の有効性は否定できない。なぜなら、逆の言い方をすれば、有効な用途が明らかであり、その用途で利用する上では確実にデータ圧縮より高い利用効率を達成できるからである。

我々は、高信頼な大規模ストレージを安価に構築するために、VLSD(Virtual Large-Scale Disks)[4)]というツールを開発してきた。本論文では、VLSD に重複排除を実現する仮想ディスクを実装し、その評価を行う。重複排除を行う手段は様々存在するが、不適切な場合はかえって容量が増加することもある。本論文では容量効率を主目的とした実装を行う。さらに、重複排除の用途として最も効果的と考えられるバックアップを想定する。通常のディスクより性能が低いことは自明であるから、容量効率に重点を置いた評価を行う。

本本文の構成は以下の通りである。2 節で関連研究として重複排除について述べる。3 節では VLSD について述べる。4 節では、VLSD における重複排除方式について述べる。5 節では、その評価を行う。最後に結論を述べる。

Vol.2012-DPS-150 No.39
Vol.2012-CSEC-56 No.39
2012/3/1

2. 関連研究

重複排除とは

データをある大きさに区切り、その中から重複部分は排除され、同一データだけを保存する容量効率を高めるための仕組みのことである。

重複排除のデータ単位

重複排除を行うデータの単位には大きく分けてファイル、ブロック、可変ブロックがある。ファイル単位では、ファイル同士の比較で重複排除を行う。ファイルの一部だけが異なると違うファイルとして保存される。ブロックはデータをブロックと呼ばれる数百 MB～数 KB 程度の大きさに区切り、そこから重複部分が排除されて保存される。ブロックサイズが一定であるということは、区切る場所によっては重複排除効率が低下する可能性がある。可変ブロックではより重複部分が増加するようにブロックサイズを変化させることができる。

重複排除方式の分類

インライン方式は、データ保存前に重複排除を行う方式である。重複排除を行うための無駄なバッファ領域を使用せずに済むため、ディスクスペースを効率よく使用できるが、リアルタイムで処理を行うため CPU の性能によっては書き込み速度が低下する恐れがある。

ポストプロセス方式は、データ保存後に重複排除を行う方式である。データ保存のためのスペースを使用しなければならないが、重複排除のタイミングを自由に決めることができる。また、書き込み速度がインライン方式よりも高速である。

重複排除製品の比較

重複排除が採用されている製品は数多く存在する。一般的なのがストレージである。主にバックアップ向けの製品が多い。重複排除のほかにさまざまな機能が搭載されているものが存在し、ユーザの用途に応じて選択することができる。また、独自のアルゴリズムを搭載したものもあり、より重複排除効果が高められている。ただし、一般的なストレージよりも高価である。ソフトウェアでも重複排除が可能な製品もある。ソフトウェアならば、どのストレージでも書き込むことができる。また、安価なストレージもにも使用することができるため、コスト削減になる。

3. VLSD

本節では大規模ストレージ構築のための VLSD(Virtual Large Scale Disk)ツールキットについて述べる。VLSD は大規模ストレージ構築のためのツールキットであり、Java によるソフトウェア RAID 実装と NBD 実装を含む。VLSD は 100% pure Java であり、Java が動作するプラットフォームの上なら VLSD も動作する。そのため Windows や Linux が混在する環境に適している。

VLSD を用いると OS に制約されることなく NBD デバイスと RAID を自由に組み合わせることができる。最低限必要な NBD デバイスはファイルサーバの 1 つである。

Linux の nbd-server コマンドや Windows の nbdsrvr コマンドは単一ファイルを仮想ディスクとして公開する。そのため 4GB の制約がある FAT32 で動作させた場合、120GB/2GB=60 プロセスの NBD サーバを稼動させる必要がある。VLSD は複数のファイルを単一の JBOD にまとめて公開することができる。

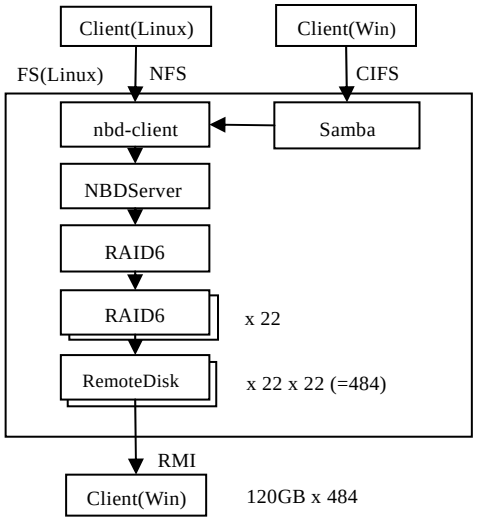

図 1. VLSD のシステム概要
Figure 1. The system overview of VLSD

ただし、VLSD の NBD サーバを用いた場合、ポート数の制約がある。ディスクを利用している最中はコネクションを維持するため NBD デバイスごとにポートを 1 つ消費する。ポート数はデバイス数より大きいため余裕があるが、その資源は無限ではない。数千台までは直接構成可能であるが、それを超える場合は間接的に、階層的に構成する必要がある。また、意図的に負荷を分散するために階層化することもある。こ

Vol.2012-DPS-150 No.39
Vol.2012-CSEC-56 No.39
2012/3/1

の問題を解消するためにポート数に制限されない RMI を用いたディスクサーバも用意した。

図 1 に VLSD を用いて分散ストレージを構成した例を示す。クライアントは 500 台存在し、その OS は Linux または Windows である。それらはそれぞれ NFS、CIFS で 1 台のファイルサーバと通信する。クライアントは同時に NBD サーバでもある。各クライアントでは空き容量を束ねた 1 つの NBD サーバが稼動する（従来のシステムでは複数の NBD サーバを稼動させなければならない場合があった）。ファイルサーバは Samba の稼動する Linux マシンである。ファイルサーバでは、クライアントの分だけ NBDDisk（後述）を作成し、22 の NBDDisk から 1 つずつ合計 22 の RAID6 を作成し、最後に 22 の RAID6 から 1 つの RAID6 を作成する。この RAID0 を NBD サーバで公開し、自分自身の NBD デバイスで参照する。

VLSD ツールキットには以下のクラスが含まれる。

Disk

すべての仮想ディスクのインターフェースを規定する。

FileDisk

単一ファイルによる固定容量ディスク。論理的な容量と物理的な容量は正確に一致する。java.io.RandomAccessFile により実装される。

VariableDisk

単一ディスクにより容量可変ディスクを作成するラッパー。8KiB を単位とする 1K 分木で管理する。葉ノードには 8KiB のデータが格納される。中間ノードには 1024 個の 64b(8B)ポインタが格納される。ノードは必要に応じて割り当てられる。 6 階層で 8EiB-1 まで拡張できる。データ以外の管理情報が保存されるため物理的な容量は 0.1% 増加する。容量可変ディスクを実現するため、Disk インターフェースには容量を追加する API が定義されている。

NBDDisk/NBDServer

NBD デバイスのクライアント。NBDServer と NBD プロトコルで通信する。その他の NBD サーバ実装（例えば、nbdsrvr）とも通信できる。

RemoteDisk/RemoteServer

遠隔デバイスのクライアント。RMI プロトコルで通信する。RemoteDisk に対応するサーバは DiskServer である。

SecureRemoteDisk/SecureRemoteServer

アクセスキーによる安全な遠隔デバイスのクライアント。RMI プロトコルで通信する。SecureRemoteDisk に対応するサーバは SecureDiskServer である。

WebDisk

Web サーバの資源を遠隔デバイスとして利用する仮想ディスク。WebDisk は、Web サーバで動作する REST 型 Web サービスにアクセスする。

JBOD

複数のディスクを直列に連結したディスク。冗長性がなく、容量増のために用いられる。各ディスクの容量は一様でなくてもよい。ストライピングを行わないため容量は単純に総和となる。例えば、100GB, 120GB, 160GB を連結すると 100+120+160=380GB になる。JBOD に対して連続的に逐次アクセスすると特定の部分ディスクに負荷が集中する。

RAID*n* (*n*=0,1,3,4,5,6)

各 RAID クラスの実装。RAID0 は HW RAID と異なり、JBOD と有意な差はない。RAID4, 5 は 1 耐故障である。RAID5 は HW RAID と異なり、RAID4 との有意な差はない。RAID6 は 2 耐故障である。P+Q 方式を採用している。

RAID4PQ/RAIDq

RAID6 と同様に 2 つのパリティ P と Q を持つ 2 耐故障 RAID である。しかし、RAID6 と異なり、RAID4 のようにパリティを専用ディスクに格納する。

RAID DP/RAIDd

水平パリティ(row parity)に加えて対角パリティ(diagonal parity)を持つ。RAID6 と同様に 2 耐故障である。2D-XOR 方式とも呼ばれる。NetApp 社の製品に使われている。2 つのパリティは互いに独立している。RAIDd を構成するにはディスク数 N は素数+1 でなければならない。

FaultDisk

耐故障性評価をおこなうためのクラス。一種のプロキシーであるが、故障を設定すると擬似的に故障を発生させる。

VotedRAID1

RAID1に似ているが，多数決で任意故障をマスクする．書込み操作はすべてのディスクに複製される．読み取り操作はすべてのディスクに複製され，その結果を多数決する．多数決のため最低3台のディスクを必要とする．稼動ディスクが 2 台以下になると正しく多数決できなくなる．

NaryRAID

クラスNaryRAIDはRAIDのサブクラスでNaryRAIDの実装である．用意された要素ディスクから指定した基数とレベルからデータディスクの台数とパリティディスクの台数を求め，ディスク番号0からデータディスクとして，データディスクの最後の番号に1足したディスク番号からパリティディスクとしてNaryRAIDを構築する．

SingleRAID

SingleRAIDは単一ディスクとRAIDとして扱うアダプタである。要素ディスクが 1 つしかないRAID0とも考えられるが、すべての作業を要素ディスクへ委譲するプロキシ

Vol.2012-DPS-150 No.39
Vol.2012-CSEC-56 No.39
2012/3/1

ーとして動作する点が単なるRAID0とは異なる。

StripeDisk

StripeDiskは、あるストライプグループ中から指定したブロックを抽出する。これによりRAIDの隠ぺいを迂回することができる。例えば、{D0,D1,D2}からなるRAID4に対してD1のみアクセスするには、ストライプグループを2とし、そのオフセットを1とすればよい。ただし、StripeDiskはあくまでRAIDを介して要素ディスクにアクセスするため、パリティディスクD2を直接読み取ることはできない。

AlternateDisk[14)]

動的に実装を切り替える。n 個の要素ディスクを持つ。要素ディスクへの要求が成功するまで順に要素ディスクを変えて試みる。すべての要素ディスクが失敗すれば全体として失敗する。

MeshRAID(44,55)

2次元直交RAIDである。直交RAIDは要素ディスクを共有する階層RAIDである。階層RAIDに比べて配線故障に強い。サブクラスにはRAID44で構成されるMeshRAID44とRAID55で構成されるMeshRAID55がある。

MeshRAID3D(444,555)[15)]

3次元直交RAIDである。直交RAID次元数は階層RAIDの階層数に該当する。次元数が増すほど耐故障性が大きくなるが、容量効率が小さくなる。サブクラスにはRAID444で構成されるMeshRAID444とRAID555で構成されるMeshRAID555がある。

GZipFolderDisk[19)]

データを圧縮して保存する仮想ディスクである。標準では1MBの比較的大きなブロックを圧縮し、個別のファイルとして分割して、格納する。符号化方式としてGZipを採用している。

4. VLSD における重複排除

ここでは、VLSD における重複排除の仮想ディスクについて述べる。重複排除にはインライン方式とポストプロセス方式がある。本論文では、VLSD の他の仮想ディスクと組み合わせることが容易なインライン方式を採用する。

我々は DedupFolderDisk を実装した。DedupFoderDisk の構造を図 2 に示す。この仮想ディスクでは、同一内容のブロックを一つのフォルダに集約して、格納する。フォルダを内容に応じて分類するためハッシュ値を用いる。今回はハッシュ値として java.util.Arrays#hashCode を用いた。このメソッドは配列の等価性の必要条件である。64 ビット長のディスクサイズに対して 32 ビットのハッシュ値は小さいが、TB サイズのディスクを扱う範囲では実用的である。64 ビットハッシュ値への拡張は今後の課題である。しかし、64 ビットより大きなハッシュ値は過大である。

ハッシュ値の重複を前提とするため、ハッシュ値の他に一意な局所 ID を生成し、両者を組み合わせて主キーとする。ハッシュ値に対してフォルダが生成される。ハッシュ値が等しくてもブロック自体が異なる場合、異なる局所 ID が割り振られる。ブロック番号と主キーの対応は index ファイルに記される。0 は番兵であり、未使用を意味する。

ブロックには逆参照が付記される。ブロックの内容が変更されたときには、元のブロックを消去してから、新たなブロックが追加される。逆参照は元のブロックを求めるために必要とされる。

Index の参照は O(1)である。ハッシュフォルダ内におけるブロックの検索は逐次的かつ線形になされるため O(n)である。ただし、ハッシュの衝突が十分に小さければ O(1)とみなすこともできる。逆参照の検索はソート済みであることを前提として二分探索可能であるため O(log n)であるが、全体を読み込むには O(n)となる。重複の度合いが高いほど逆参照の検索には時間がかかる。

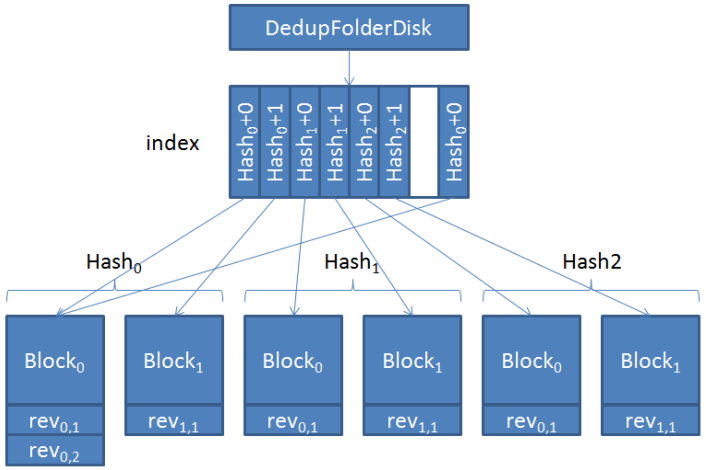

図 2. DedupFolderDisk の構造
Fig.2 The structure of DedupFolderDisk

DedupFolderDisk における読み取りは以下のように行われる。Index からブロック番号に対するハッシュ値と局所 ID を求める。これらの値を用いてブロックファイルが

Vol.2012-DPS-150 No.39
Vol.2012-CSEC-56 No.39
2012/3/1

一意に決まる。ブロックファイル中に読み取るべきブロック番号が逆参照として存在することは前提であるが、自明であるので、単にブロックを返せばよい。

同じく書き込みは以下のように行われる。Index からブロック番号に対するハッシュ値と局所 ID を求める。これらの値を用いてブロックファイルが一意に決まる。ブロックファイル中の逆参照からブロック番号を削除し、index の対応する箇所を一旦 0 にする。しかる後、新たなブロックの内容に応じてハッシュ値を求め、対応するフォルダに同じブロックがないか確認する。あれば逆参照に追加し、なければ新たな局所 ID を割り当て、ブロックファイルを作成する。

最後に、実装に関する考察を述べる。重複排除は Map を用いて実装できる。永続的なオブジェクトをサポートする DB のクラスライブラリには StoredMap などもあるが、これらを用いた実装が必ずしも容量効率がよいわけではない。例えば、典型的な DB ライブラリでは B 木を用いているが、B 木の容量効率は最悪の場合 50%になる。

5. 評価

ここでは、重複排除ディスクの評価を行う。評価基準は性能、圧縮率である。圧縮率は論理容量に対する物理容量の比率である。

ここでベンチマークは 400 回のランダムな読み書きである。評価環境は以下の通りである。OS: Windows 7 Pro 64bit SP1, CPU: Intel Core-i7 U640 1.2GHz, Mem: 2GB, SSD: 128GB。これらは以後のすべての評価で変わらない。

はじめに性能を評価する。まず、もっとも単純な仮想ディスクである FileDisk と性能を比較する。結果を表 1 に示す。ただし、DedupFolderDisk のブロックサイズは 8KB とした。この結果から DedupFolderDisk の性能は非常に遅いと言える。これは純粋にソフトウェア処理であることと実装が十分最適化されていないことが原因である。しかし、DedupFonderDisk は 1 度の書き込みにおいて 4 回の書き込みを行うため、この差は埋めがたい。この点では表 1 の結果は I/O がある程度隠ぺいされていると言える。

表 1. DedupFolderDisk vs FileDisk
Table 1. DedupFolderDisk vs FileDisk

Class	Benchmark[s]
FileDisk	0.905
DedupFolderDisk	2.356

性能はブロックサイズに依存する。次にブロックサイズを 512B～8KB まで変化させたときの、それぞれのベンチマークを表 2 に示す。さらに、各ブロックサイズの重複排除の圧縮率を表 3 に示す。これらの結果からブロックサイズが比較的大きい方が性能は高くなると言える。一方、ブロックサイズが小さい方が重複排除の圧縮率は高くなると予想されるので、ブロックサイズの選択は用途に依存したトレードオフであると考えられる。

表 2. ブロックサイズに依存する性能
Table 2. Performance depended on Block Size

Block size[B]	Benchmark[s]
512	22.354
1024	9.875
2048	5.101
4096	3.074
8192	2.356

表 3. ブロックサイズによる重複排除の効果
Table 3 .Deduplicated ratio depended on Block Size

Block size[B]	Deduplicated Ratio
2048	26.0%
4096	26.0%
8192	27.2%

本論文では、バックアップを主な用途とする。また、本システムは教育環境での利用を前提に開発されているため、バックアップの対象は教育用クラウドのイメージとする。そこで、単一イメージについて重複排除による圧縮率を評価する。なお、ここで教育用クラウドのイメージの OS は Ubuntu 11.10 とする。表 3 に 4GB イメージを用いた場合の圧縮率を示す。圧縮率はブロックサイズが小さい方が高いことが示された。

表 4. 重複排除の効果(1)
Table 4. The effect of deduplication (1)

Image size[GB]	Deduplicated size[GB]	Ratio
4GB	1.1GB	27.2%
8GB	1.2GB	14.3%
12GB	0.6GB	5.4%

基本的にはインストール直後の各ブロックには重複する要素はほとんどないと考えられる。よって、重複排除の効果は未使用領域の大きさに比例する。表 4 に結果を

Vol.2012-DPS-150 No.39
Vol.2012-CSEC-56 No.39
2012/3/1

示す。未使用領域の大きい 12GB、8GB、4GB の順で圧縮率が高いことが分かる。

複数のユーザが同じデータを保存しているとき重複排除の効果が高い。仮想マシンのイメージには多くの共通するデータがあるため重複排除の効果は高いと考えられる。そこで、ある仮想マシン X の内部に別の仮想マシン A, B のイメージを保存し、X 全体を重複排除する。このとき A, B のイメージは等しいものとする。X, A, B それぞれ OS に 4GB を割り当てていると仮定し、A, B のイメージサイズを 4GB、X のイメージサイズを 12GB(=4GB+2*4GB)とする。予想される圧縮率は 5%であるが、実際はその３倍になっている。これは、ブロックサイズが大きすぎたためである。

表 5. 重複排除の効果(2)
Table 5. The effect of deduplication (2)

Image size[GB]	Deduplicated size[GB]	Ratio
12GB	1.7GB	14.0%

一般的に N 個の同じイメージを含む 4+4N[GB]のイメージは 4+4[GB]に重複排除される。よって、圧縮率は(4+4)/(4+4N)=2/(1+N)となる。

表 6. 重複排除の効果(3)
Table 6. The effect of deduplication (3)

Image	Deduplicated size[GB]	Ratio
Before update	1.1GB	27.2%
After update	1.2GB	28.8%

バックアップでは、異なる版を保存することがある。そこで、インストール直後のイメージとパッケージ更新後のイメージを比較する。先程の評価で B の代わりに B を更新した B’を用いる。結果を表 6 に示す。表 4 に対する deduplicated size の差がパッケージ更新の差分である。よって、相当なパッケージ更新がなされても、そのイメージ全体への影響は高々9%であると言える。

6. まとめ

本論文では、重複排除の効果についての評価を行ったが、この結果は容量削減という点で十分なものであるといえる。4GB と 8GB のイメージでは、サイズが違うにもかかわらず重複排除後の容量がほぼ等しい。これは、イメージの未使用領域が重複排除されていると考えられる。また、ブロックサイズと性能のトレードオフの問題は依然として残るが、バックアップという限定的な用途での使用であるならば、これは許容の範囲内であるといえる。

今後の課題として、本論文で提案した DedupFolderDisk では、ハッシュ値を求めるために hashcode を用いた。ハッシュ値を求めるためのアルゴリズムはこのほかにも SHA や MD5 などがある。このようなアルゴリズムで重複排除を行ったときに性能にどのような影響がでるのかを検証する必要がある。また、速度と性能のトレードオフの関係は固定長ブロックから可変長ブロックに変更することで改善できると考えられる。

参考文献

1) Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson: “RAID: High-Performance, Reliable Secondary Storage,” ACM Computing Surveys, Vol. 26, No. 2, pp.145-185, June 1994
2) P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, and D.A. Patterson: “RAID: High-Performance, Reliable Secondary Storage,” ACM Computing Surveys, Vol. 26, No. 2, pp.145-185, June 1994
3) S. Hoon Baek, B. Wan Kim, E. Joung Joung and C. Won Park: “Reliability and Performance of Hierarchical RAID with Multiple Controllers,” In Proc. of 20th annual ACM symposium on Principles of Distributed Computing, pp.246-254, (2001)
4) Erianto Chai, Minoru Uehara, Hideki Mori, Nobuyoshi Sato: "Virtual Large-Scale Disk System for PC-Room", LNCS 4658, Network-Based Information Systems, pp.476-485, (2007.9.3-4)
5) Katsuyoshi Matsumoto, Minoru Uehara: "N-nary RAID: 3-resilient RAID based on an N-nary number", In Proceedings of 23rd International Conference on Advanced Information Networking and Applications(AINA2009), pp.249-255, (2008.5.26)
6) Minoru Uehara: "Combining N-ary RAID to RAID MP", In Proc. of 1st International Workshop on Information Technology for Innovative Services(ITIS2009) in conjunction with 2009 International Conference on Network-Based Information Systems(NBiS2009), pp.451-456, (2009.8.19-21)
7) Yuji Nakamura, Minoru Uehara: "Improving the performance of N-ary RAID by writing with XOR operation", In Proc. of 2011 25th IEEE International Conference on Advanced Information Networking and Applications (AINA2011), pp.633-638, (Biopolis, Singapore, 2011.3.22-25)
8) Minoru Uehara, Makoto Murakami, Motoi Yamagiwa: " A Proposal of 3 FT Orthogonal RAID and Its implementation in Virtual Large-Scale Disk ", IPSJ Journal Vol.52, No.2, pp.434-445, (2011.2) (in Japanese)
9) Minoru Uehara: "A Toolkit for Virtual Large-Scale Storage in a Learning Environment", In Proc. of 21th International Conference on Advanced Information Networking and Applications Workshops/Symposia 2007, Vol. 1, pp.888-893, (2007.5.23)
10) Minoru Uehara: "Composite RAID for Rapid Prototyping Data Grid", International Journal on Web and Grid Service, Vol.7, No.1, pp.58-73,(2011.1)
11) Minoru Uehara: "3 Faults Tolerant Orthogonal RAID for Large Storage", In Proc. of 2010 International Conference on Network-Based Information Systems(NBiS2010), pp.209-215, (2010.9.14-16, Gifu, Japan)

Vol.2012-DPS-150 No.39
Vol.2012-CSEC-56 No.39
2012/3/1

12) Yuji Nakamura, Minoru Uehara: "An Implementation of NaryRAID", In Proc. of 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA2010), pp.134-141, (Perth, Australia, 2010.4.20-23)
13) Yuji Nakamura, Minoru Uehara: "Performance Evaluation of 2FT RAID", In Proc. of 3rd International Workshop on Information Technology for Innovative Services(ITIS2011) in conjunction with the 14th International Conference on Network-Based Information Systems(NBiS2011), pp.529-534, (2011.9.7-9,Tirana,Albania)
14) Minoru Uehara: "An Alternative Implementation of 3FT RAID in Virtual Large Scale Disks", INCoS2011, (TBA)
15) Minoru Uehara: "Design and Implementation of 3D MeshRAID in Virtual Large-Scale Disks", MNSA2011, (TBA)
16) James S. Plank: "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems", SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 27(9), 995–1012 (SEPTEMBER 1997)
17) "圧縮率比較一覧表", http://www.emit.jp/gca/cmptest.html
18) Alan Radding: "重複排除技術の俗説と方法論", Storage Magazine 2008 年 9 月号, http://www.jdsf.gr.jp/backup/storage_theory.html
19) 柴崎 敬大, 上原 稔: "圧縮ファイルを用いた仮想ディスクの研究", 第 73 回情報処理学会全国大会, 3X-9, pp.383-384, (2011.3.2-4, 東工大)