社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－DBS－143（90）
2007／7／4

性能の異なるディスクが混在する環境でのデータ配置及びディスク配列に関する考察

片居木 誠† 小林 大† 横田 治夫†,††

† 東京工業大学 大学院情報理工学研究科 計算工学専攻 〒152-8552 東京都目黒区大岡山 2-12-1
†† 東京工業大学 学術国際情報センター 〒152-8552 東京都目黒区大岡山 2-12-1
E-mail: †{mkataigi,daik}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし 我々は，多様な性能・容量のディスクを含む混在型ストレージシステムにおいて，システム全体の帯域及び容量の利用率の向上を目指している．これまでに均一環境でのアクセス負荷とデータ容量を均衡化させる手法としてAOD法を提案してきたが，本稿では混在環境で帯域と容量の比率(BSR)を均衡化するようにAOD法を拡張する．その場合のデータ配置条件と性質の異なるディスクの配列方法に関して考察する．
キーワード ストレージシステム，負荷均衡化，容量均衡化

Consideration of data placement and disk sequencing in heterogeneous disk environment

Makoto KATAIGI†, Dai KOBAYASHI†, and Haruo YOKOTA†,††

† Dept. of Computer Science, Grad. School of Info. Sci. and Eng, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8552 Japan
†† Global Scientific Information & Computing Center, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8552 Japan
E-mail: †{mkataigi,daik}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

Abstract We aim to improve the utilization ratio of both the system bandwidth and capacity in heterogeneous disk environment. AOD is a method we proposed to balance both access load and data capacity. We extend the method for balancing Bandwidth to Space Ratio (BSR) in the heterogeneous disk environment. We consider the conditions of data placement and disk sequencing.
Key words Storage systems, Load balancing, Capacity balancing

1. まえがき

近年コンピュータシステムで保存されるデータ量が増大している．このため，多数のハードディスクドライブ(HDD)を並列接続させたストレージシステムが用いられている．

大規模ストレージシステムでは，アクセス負荷やデータ量の均衡化を行う必要がある．これは，例えばHDDではアクセス量が増加するとレスポンスタイムが悪化するため，システム内の一部のHDDにアクセスが集中するとシステム全体としての平均レスポンスタイムが悪化するためである．また，データが冗長化されている環境では故障回復戦略において，各HDDに保存されているデータ量が均衡化されていないと，どのディスクが故障するかによって復旧対象のデータの量がばらついてしまい，非効率になってしまうためである．

アクセス負荷の均衡化戦略とデータ量の均衡化戦略を同時に用いる場合，それぞれの戦略のデータ操作が相反する操作となることが起こり得るという問題があった．この問題に対して我々は両戦略を両立するための方法として，AOD(Adaptive Overlapped Declustering)法[1]を提案してきた．これはデータ冗長化のためのバックアップを保持している環境で，そのプライマリデータとバックアップデータのアクセス傾向の違いを，アクセス・データ量均衡化の両立に利用した戦略である．

ストレージシステムではディスクが故障や老朽化した時にディスク交換や，システムの性能や容量が不足した時にディスク追加を行う．またシステムを構成する個々のHDDも急速に高性能化・大容量化している[2]．しかし従来のシステムではディスク交換・追加に対して旧ディスクと同じ型のディスクを使用したり，新しい高性能・大容量であるディスクの性能・容量を他の古いディスクの性能・容量にあわせて制限するなど，新しいディスクのリソースを有効に利用できていなかった．

この理由としては，個々のディスクの性能や容量が違う環境(混在環境)においては前述の単純なアクセスやデータ量の均衡化は難しいことがあげられる．これは個々のディスクの性能・容量が違うため，各ディスクのアクセス量を均衡化させても，各ディスクのデータ転送性能が均衡化されるわけではなく，また保存データ量を均衡化させても各ディスクの容量利用率が均衡化されないためである．

ここで混在環境においては，性能対容量比であるBSR(Bandwidth to Space Ratio)がシステム内の各ディスクにおいて均衡化している場合は個々のディスク性能や容量を有効利用できるという研究結果がある [3]．しかし，実際には個々のディスクのBSRは必ずしも等しくない．

本研究では，各ディスクの性能・容量を有効に利用できていないという問題に対し，AOD法を拡張し全ディスクのプライマリデータに対するBSRを均衡化して解決する．つまり，プライマリデータの配置によって性能(BSR)の均衡化，バックアップの配置によって容量の均衡化を行う．本稿では，特にバックアップを保持して配置できるデータの容量を最大化するために必要な条件，及び各ディスクの領域をプライマリ・バックアップデータを配置すべき領域に分割する方法について示す．

以下にこの論文の構成を示す．2. で関連研究を示す．3. でAOD法の簡単な説明について記す．4.1 でシステムの容量についての考察を行い，4.2 でディスクの配列についての考察を行い，4.3 でBSRににについての考察を行う．

2. 関連研究

近年，混在環境でのデータ配置に関する多くの研究が行われている．例えば，ストライピングやRAIDを混在環境で実現する研究が行われている [4–6]．これらの研究ではそれぞれのディスクの性能と容量の比(BSR)がほぼ一定，または正の相関関係があると仮定して，それぞれのディスクに配置するデータユニットの量をディスク容量に比例させるように変更させることによって実現している．しかし，実際にはディスクの性能と容量には必ずしも正の相関があるわけではない．

ストライピングやRAID以外でも混在環境でのデータ配置に関する多くの研究が行われている．[7] では，複数のディスクを仮想的に一つのディスクとみなす，または一つのディスクを仮想的に複数のディスクに分割することによってそれぞれの仮想ディスクのBSRを一定にすることによって，システム性能とシステム全体の利用可能容量を向上させている．しかし，ここでは必ずしも仮想ディスクのBSRを一定にすることが出来ない場合も存在するという問題点が存在する．

[8] では，各データへのアクセス頻度の違いを利用して，混在環境においてシステム全体の性能と利用可能容量を向上させている．しかし，この方法は事前にデータのアクセス傾向が得られている場合でないと適用が難しく，また事前にデータのアクセス傾向が分かっているシステムというのは一般的でない．

[3] では，各ディスクのBSRに注目し，それぞれのディスクのBSRとそのディスク上のデータのBSRの乖離度によって配置場所を決める方法である．実験では各ディスクのBSRの値が均衡化しているときは利用できない帯域や容量が少なくなるという結果が得られている．しかしBSRの値が均衡していない環境においては結果は芳しくない．

この様に，システムを構成する各ディスクのBSRが等しい環境では良い結果が得られている．しかし，実際には各ディスクのBSRは必ずしも一定ではない．そのため，いかに各ディスクのBSRが等しいシステムを作るかが課題となっている．

3. AOD 法

ストレージシステムではデータのアクセスパスを確保するためにインデックス構造が用いられる．インデックス構造の中で値域分散法はハッシュ法などに比べ，レンジクエリを利用可能などアクセス性能が良いためしばしば用いられる．また多くのストレージシステムではデータに冗長性を持たせるためにバックアップを生成し，またこのバックアップの配置にはCD(Chained Declustering)法 [9] による配置法を用いることがある．しかし，この配置法においてはバックアップの配置が一方向のみであるため，アクセス負荷と容量バランスの均衡化を同時に行うのが難しい．

AOD(Adaptive Overlapped Declustering)法ではバックアップ配置可能位置を左右両方向にし，データ量の少ないディスクに配置することにより，プライマリデータによるアクセス負荷の均衡化とバックアップデータによる容量バランスの均衡化を同時に達成している．

AOD法は，全アクセスはプライマリデータに対して行われる状況において，プライマリデータとバックアップデータのアクセス頻度に差が出ることを利用した方法である．そのため，混在環境のようなシステムを構成する各ディスクの性能と容量が違う場合においても，プライマリデータとバックアップデータのアクセス量の違いを利用し，データの配置を決定することにより，それぞれを均衡化することが出来る．

4. 混在環境へのAOD法の拡張

AOD法はプライマリデータを配置したディスクの左右のディスクにバックアップデータを振り分ける戦略であるが，それぞれのディスクの容量が等しくない場合にも拡張可能である．まず，4.1 ではバックアップデータの左右の割り当てによる各ディスクの容量の差への対応を考察する(図1)．4.2 では，このシステムでバックアップデータを配置可能な領域の容量(利用可能容量)を大きく取るためのディスクの配列について考察する．

このとき，プライマリデータとそのバックアップデータは同じサイズであればよいので，それぞれの領域(図1中の R_{n-1} 及び L_n)に配置するプライマリデータ量を調整することによって，性能の差に対応できる(図3)．4.3 では，BSRを均衡化させるためのプライマリデータの配置について考察する．

特に本稿では，右隣・左隣のディスクと対応する領域，プライマリデータ・バックアップデータを配置する領域とそれぞれの領域に分けて考える．各領域への実際のデータ割り当ては今後の課題である．

なお本文中の変数の意味は表1に示す通りである．

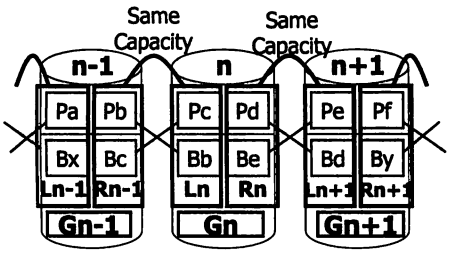

図 1　隣接ディスクの容量の関係

表 1　ディスク n における変数定義

B_n	帯域
C_n	容量 ($C_n = L_n + R_n + G_n$)
D_n	利用可能容量 ($D_n = L_n + R_n$)
L_n	対応する部分をディスク $n-1$ に持つ領域の容量
R_n	対応する部分をディスク $n+1$ に持つ領域の容量
G_n	ディスク $n-1$，ディスク $n+1$ のどちらにも バックアップを配置できない領域の容量
P_n	配置されるべきプライマリデータの容量
P_{L_n}	領域 L_n に配置されるプライマリデータの容量
P_{R_n}	領域 R_n に配置されるプライマリデータの容量

※ N：全ディスク台数 [1..N]

表記の簡便のため，ディスク $n(> N)$ となるディスクについては適宜ディスク $n \bmod N$ で読み変える．また $n(\leqq 0)$ についても $|n| < \alpha N$ となる整数 α を用い，ディスク $(n + \alpha N) \bmod N$ で読み変える．

4.1　システムの利用可能容量の最大化

バックアップデータを配置不可能な領域が増えることは非効率である．混在環境に CD 法によるバックアップ配置を適用した場合，バックアップデータを配置不可能な領域が生成されてしまう可能性がある．それに対して AOD 法を適用した場合，バックアップデータを配置不可能な領域は少なくとも CD 法より小さい領域にすることが可能であり，無くすことができる可能性もある．よって AOD 法を使い，さらにシステム内の利用可能容量を最大化することを考える．

あるディスク内のプライマリデータを，バックアップを右隣のディスクに配置するデータと左隣のディスクに配置するデータに分けて考えるとする．このとき，プライマリデータとそれに対応するバックアップデータは同じサイズになるので，結局連続するディスク間で同じ容量の領域が発生する (図 1)．よって，連続するディスク間は同じサイズの領域が与えられなければならない．またこれらの領域は 0 以上でなくてはならない．これより各変数が満たすべき条件である次の式 (1) が得られる．

$$R_n \geqq 0, \quad L_n \geqq 0, \quad R_{n-1} = L_n \tag{1}$$

この条件の元，利用可能容量を最大化することを考える．つまり，以下の式 (2) の D_{sum} を最大化することを考える．

$$D_{sum} \equiv \sum_{k=1}^{N} D_k \qquad (\text{ただし } D_n = L_n + R_n) \tag{2}$$

4.1.1　D_n 決定のための満たすべき条件

式 (1) はディスク $n+m$ の L_{n+m} とディスク n の $(-1)^m L_n$ の差 δ_c を用いて式 (4) ように変形される (付録 1.)．以下の式中の M_{odd} は 1 以上 N 以下の奇数，M_{even} は偶数の集合とする．

$$\delta_c(n,m) \equiv L_{n+m} - (-1)^m L_n = \sum_{k=1}^{m} (-1)^{k-1} D_{n+m-k} \tag{3}$$

$$max_{x \in M_{even}}(-\delta_c(n,x)) \leqq min_{y \in M_{odd}}(\delta_c(n,y)) \tag{4}$$

ここで，式 (4) はある n について成り立つとき，他のあらゆる n についても成り立つ (付録 2.)．よって，どの n を用いても同じであるので，以降は $n = 1$ とする．このとき $\delta_c(m) = \delta_c(1,m)$ とし，式 (4) を以下の式 (5) と書きなおす．

$$max_{x \in M_{even}}(-\delta_c(x)) \leqq min_{y \in M_{odd}}(\delta_c(y)) \tag{5}$$

ディスク台数は N であるため，$L_{1+N} = L_1$ が成り立つ．よって境界条件として，式 (6)・式 (7) が得られる (付録 3.)．

$$\delta_c(N) - 2L_1 = 0 \quad (N \in odd) \tag{6}$$

$$\delta_c(N) = 0 \qquad (N \in even) \tag{7}$$

また，付録 4. より，$G_m = G_m + \Delta$ に値を変化させると，n から $n+m-1$ までのディスクの $\delta_c(m)$ は変化せず，$n+m$ から $n+N$ までの $n+m+x(x \in M_{odd})$ の $\delta_c(m)$ は $+\Delta$，$n+m+x(x \in M_{even})$ の $\delta_c(m)$ は $-\Delta$ となる．

4.1.2　領域容量決定アルゴリズム

ある m について，$\delta_c(m)$ が式 (5) を満たしていない場合，m が奇数の場合は $\delta_c(m)$ の値を増やしたい，また m が偶数の場合は $-\delta_c(m)$ を減らしたい．よって，m が奇数の場合も偶数の場合も結局 $\delta_c(m)$ を増やしたい．しかし，G_m の値を増加させても $\delta_c(m)$ の値は減少してしまう．このため，$\delta_c(m)$ を増やすためには，$l = m - x(x \in M_{odd})$ の G_l を増やす必要がある．

また，最後の $m = N$ については境界条件があるため，この境界条件を満たすように $\sum_{k=1}^{N} G_k$ を調整しなければいけない．

以上よりデータを配置する前に，利用可能容量を最大化する際に，D_n を決定するアルゴリズムは表 2 の通りとなる．

また，式 (1)・式 (A·1) より $L_n \cdot R_n$ を計算することができる．

4.2　利用可能容量最大化のためのディスクの配列

利用可能容量を最大化する際の式 (4) における $\delta_c(m)$ はディスクの配列によって変わってくる．そのため，利用可能容量を最大化するには，ディスクの配列も同時に考える必要がある．

この節ではこれ以降，容量 D_n の配列について考える．

4.2.1　ディスク容量配列の増減と容量の条件の変化

D_n の増減と $\delta_c(m)$ の関係について考察する．$\delta_c(m)$ の変化量は次の式 (8) ようになる．

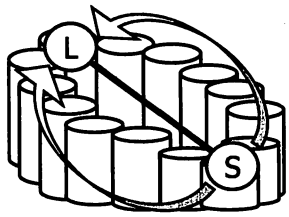

図 2　切り株型の配置

表 2　利用可能容量最大化のための利用容量決定アルゴリズム

Input: ディスク容量の配列 C_n
Output: 各ディスクのバックアップを配置可能な容量 D_n
Step1. $m=1, min=\infty, max=-\infty, \delta_c(0)=0$
Step2. $\delta_c(m)=C_m-\delta_c(m-1)$
Step3. $m \in odd$ かつ $min > \delta_c(m)$ の場合 $min=\delta_c(m)$
Step4. $m \in even$ かつ $max < -\delta_c(m)$ の場合 $max=-\delta_c(m)$
Step5.1 $m \in odd$ かつ $max < -\delta_c(m)$ の場合
$min=max, k=m-1, G_{sum}=-\delta_c(m)-max, \delta_c(m)=-max$
Step5.2. $m \in even$ かつ $min > \delta_c(m)$ の場合
$min=max, k=m-1, G_{sum}=\delta_c(m)-min, \delta_c(m)=min$
Step5.3. $G_k=G_{sum}$
Step5.4. $G_k > C_k$ の場合,
$G_{sum}=G_k-C_k, G_k=C_k, k=k-2$ として Step5.2 へ
Step6. $m=m+1$
Step7. $m<N$ の場合，Step.2 へ
Step8.1 $N \in odd$ かつ $max \leqq \frac{1}{2}*\delta_c(N) \leqq min$ でない場合
Step8.1.1. $max > \frac{1}{2}*\delta_c(N)$ の場合，$k=m-1, G_{sum}=\delta_c-max$
Step8.1.2. $max > \frac{1}{2}*\delta_c(N)$ の場合，$k=m, G_{sum}=min-\delta_c$
Step8.2 $N \in even$ かつ $\delta_c(N) \neq 0$ の場合
Step8.2.1. $\delta_c(N)>0$ の場合，$k=m-1, G_{sum}=\delta_c(N)$
Step8.2.2 $\delta_c(N)<0$ の場合，$k=m, G_{sum}=-\delta_c(N)$
Step8.3 $G_k=G_{sum}$
Step8.4 $G_k > C_k$ の場合,
$G_{sum}=G_k-C_k, G_k=C_k, k=k-2$ として Step8.3 へ

$$\delta_c(m)=D_m-D_{m-1}+\delta_c(m-2) \tag{8}$$

まず連続する $n \in [n_{start}..n_{end}]$ で D_n が増加している箇所について考える．奇数番目のディスク n について，$D_n-D_{n-1} \geqq 0$ であるので $\delta_c(n)$ は増加する．同様に偶数番目ディスク n については，$D_n-D_{n-1} \geqq 0$ であるので $-\delta_c(n)$ は減少する．これより，D_n の増加部分の開始点 n_{start} について n_{start} 以前で n_{start} と偶奇が逆である全ての点 $m \in [1..n_{start}-1]$ と比較し，$n_{start} \in M_{even}$ の場合は $\delta_c(n_start) \geqq -\delta_c(m)$，$n_{start} \in M_{odd}$ の場合は $\delta_c(m) \geqq -\delta_c(n_{start})$ を満たしていれば，$[n_{start}..n_{end}]$ 全体が式 (4) を満たす．

また逆に D_n が減少している箇所についても同様になる．結局 D_n が増加している箇所では開始点 n_{start}，D_n が減少している箇所では終点 n_{end}，つまり D_n が減少から増加に転じる箇所 (極小点) において式 (4) を満たしていれば良い．

ここで，最終ディスク N のプライマリデータに対する右隣のバックアップデータはディスク 1 に配置され，ディスク 1 のプライマリデータに対する左隣のバックアップデータはディスク N に配置される．よって，容量配列 D_n は仮想的に最終ディスク N の右隣はディスク 1，つまり循環していると考られる．配列が循環している場合，極小点は少なくとも一箇所存在する．

容量最小ディスクから見て順に昇順 → 降順となる配列 (切り株型と呼ぶ) は極小点が 1 つとなり上記の条件を満たす配列の一つである．これ以降この形を基準として考える (図 2)．

4.2.2 新規ディスクの追加

式 (6) 及び式 (7) で示した境界条件のため，$\delta_c(N)$ は 0 に近づけなくてはならない．ところで，ディスクは循環していてかつ最小ディスクから昇順 → 降順で並んでいるので，ある容量のディスクを追加しようとした場合，候補が 2 箇所存在することになる．よって，新規ディスクはその 2 箇所のうち境界値 $\delta_c(N)$ が小さい方に配置すべきである．新しいディスク (容量 D_{new}) を追加する位置を決定するアルゴリズムを表 3 に示す．

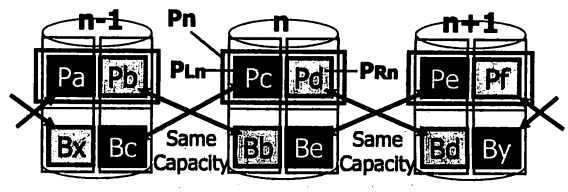

図 3　隣接ディスクのプライマリデータ量の関係

表 3　新規ディスク追加位置決定アルゴリズム

Input: 新規ディスクの容量 D_{new}，現在のディスクの容量の配列 D_n
Output: 新規ディスクを追加すべき位置
Step1. ディスク 1 から順に $D_m \leqq D_{new} \leqq D_{m+1}$ となる箇所 (m とする) と，$D_n \geqq D_{new} \geqq D_{n+1}$ となる箇所 (n とする) を探査する
Step2. m, n それぞれに新ディスクを追加した場合の $\delta_c(N)$ を計算する
Step3. $|\delta_c(N)|$ の小さい方に新ディスクを追加する．

また，これによって新ディスクを追加した状態でも，最小ディスクから順に昇順 → 降順となる配列を満たしている．よって，再帰的にこのアルゴリズムで生成される形は切り株型となる．

4.3 BSR の均衡化

システムの性能はそれぞれのディスク上に配置されている全データの容量，アクセス傾向，及びディスク自身の性能などが大きく関係するため，システム内での各ディスクの性能の均衡化は難しい．それに対して，性能対容量比である BSR(Bandwidth to Space Ratio) がシステム内の各ディスクにおいて均衡化している場合においては個々のディスク性能や容量を有効利用できるという研究結果がある [3]．しかし，必ずしも全てのディスクの BSR が均衡化しているわけではなく，また直接ディスクの BSR を変更することは出来ない．そのため，本稿ではシステム内の各ディスクのプライマリデータに対する BSR を均衡化させることを考える．これはアクセスは全てプライマリデータに対して行われるため，バックアップデータに対するアクセスは 0 であり，その BSR は常に 0 となるためである．

4.3.1 BSR を均衡化する条件

領域 $R_{n-1}+L_n$ 内でのプライマリデータとバックアップデータのサイズは一致しなければならない，つまり式 (9) を満たさなければいけない (図 3)．

$$P_{R_{n-1}}+P_{L_n}=R_{n-1} \quad (=L_n) \tag{9}$$

このとき，各ディスクのプライマリデータに対する BSR を $BSR_n \equiv B_n/P_n$ とし，式 (10) の BSR_{dis} を最小化させる．

$$BSR_{dis} \equiv max(BSR_n)-min(BSR_m) \tag{10}$$

$B_{all} \equiv \sum_{k=1}^{N} B_k$ 及び $D_{all} \equiv \sum_{k=1}^{N} D_n$ と定義する．このとき，P'_n を次の式 (11) とする．

$$P'_n \equiv \frac{1}{2} D_{all} * \frac{B_n}{B_{all}} \quad (11)$$

各ディスクにプライマリデータの領域として P'_n だけ取れるとき，つまり $P_n = P'_n$ となるとき，各ディスクの BSR_n は次の式 (12) となり一定となる．

$$BSR_n = \frac{B_n}{P_n} = 2\frac{B_{all}}{D_{all}} \quad (12)$$

よって各ディスクに P'_n だけのプライマリを配置できない場合を考える．

4.3.2 P_{L_n}, P_{R_n} を正しく割り当てるための条件

まず，P_{L_n}, P_{R_n} が正しく 0 以上に割り当てられるための条件について考察する．条件より以下の式 (13) が成り立つ．

$$P_{L_n} = R_{n-1} - P_{R_{n-1}}, \quad P_{R_n} = P_n - P_{L_n} \quad (13)$$

この式より，ディスク $n+m$ の $P_{L_{n+m}}$ とディスク n の P_{L_n} の差 $\delta_{bsr}(n,m)$ を用いて，ディスク n から右に m 台目のプライマリデータ領域として次の式 (15) 及び式 (16) が得られる．

$$\delta_{bsr}(n,m) \equiv P_{L_{n+m}} - P_{L_n} = \sum_{k=0}^{m-1} P_{n+k} - \sum_{k=0}^{m-1} R_{n+k} \quad (14)$$

$$P_{L_{n+m}} = -\delta_{bsr}(n,m) + P_n \quad (15)$$

$$P_{R_{n+m}} = \delta_{bsr}(n,m) + P_{n+m} - P_n \quad (16)$$

ここで，条件より $P_{L_n} \geqq 0, P_{R_n} \geqq 0$ であるので，次の式 (17) が成り立たねばならない．

$$\delta_{bsr}(n,m) \leqq P_n \leqq \delta_{bsr}(n,m) + P_{n+m} \quad (17)$$

条件より $P_n \geqq 0$ がある必要である．そのため，P_{L_n}, P_{R_n} が正しく割り当てられるためには，次の式 (18) が成り立たなければならない．

$$\delta_{bsr}(n,m) + P_{n+m} \geqq 0$$

$$\therefore 0 \leqq \sum_{k=0}^{m} P_{n+k} - \sum_{k=0}^{m-1} R_{n+k} \equiv \delta_{bsr}(n,m+1) \quad (18)$$

4.3.3 プライマリデータ量調整による $\delta_{bsr}(n,m)$ の変化

次に，プライマリデータ量を変化させた際の $\delta_{bsr}(n,m)$ の変化について考える．ディスク $n+m$ のプライマリデータ量 P_{n+m} を Δ だけ増やした場合，式 (19) で示すように m 以降の全ての $\delta_{bsr}(n,m)$ が Δ だけ増える．

$$\delta_{bsr}(n,m)' = \sum_{k=0}^{m} P_{n+k} - \sum_{k=0}^{m-1} R_{n+k} + \Delta = \delta_{bsr}(n,m) + \Delta \quad (19)$$

同様に，ディスク $n+m$ のプライマリデータ量 P_{n+m} を Δ だけ減らした場合，m 以降の全ての $\delta_{bsr}(n,m)$ が Δ だけ減る．よって，ディスク $n+m$ のプライマリデータ量 P_{n+m} を Δ だけ変化させると，m 以降全ての $\delta_{bsr}(n,m)$ が Δ だけ変化する．

4.3.4 プライマリデータ量決定アルゴリズム

番号 l における $\delta_{bsr}(n,l)$ の値を Δ 増やすためには，$m \in [1..l]$ の間で P_{n+m} を合計 Δ 増やせば良い．ここで，プライマリデータ量の合計は常に $\frac{1}{2}D_{all}$ である必要があるため，ある P_n について Δ だけ増やしたら，他の P_n を Δ だけ減らさなければならない．そのため，$m \in [1..l]$ 以外のディスク，つまり $m \in [l+1..N]$ の間で P_{n+m} を合計 Δ だけ減らす必要がある．

さらに，式 (10) の BSR_{dis} を最小化しなければならない．各ディスクのプライマリデータに対する BSR の上昇量，下降量がばらついている場合はばらついていない場合に比べて．最大値は大きくなり最小値は小さくなる．そのため，各ディスクの P_n の増減値は BSR_n が均等に増減するようにすべきである．

これより各ディスクのプライマリデータに対する BSR を均衡化する際に各ディスクに配置すべきプライマリデータ量を決定するためのアルゴリズムは表 4 のようになる．

表 4　P_n 決定アルゴリズム

Input: 各ディスクの右隣ディスクと対応する領域の容量 $R_n(=L_{n+1})$ 及び各ディスクの帯域 B_n
Output: 各ディスクに割り当てるべきプライマリデータの容量 P_n
Step1. $n=1$
Step2. 全ての n について，$P_n = \frac{1}{2}D_{all} * \frac{B_n}{B_{all}}$ とする
Step3. $m \in [1..N]$ について $\delta_{bsr}(n,m)$ の最小値を探し，δ_{bsr}^{min} とする
Step4. $\delta_{bsr}^{min} \geqq 0$ の場合，Step8. へ
Step5. $m \in [1..min]$ の $P_m = P_m + |\delta_{bsr}^{min}| * \frac{B_m}{\sum_{k=1}^{min} B_k}$
Step6. $m \in [min+1..N]$ の $P_m = P_m - |\delta_{bsr}^{min}| * \frac{B_m}{\sum_{k=min+1}^{N} B_k}$
Step7. Step3. へ
Step8. n を 1 増やす
Step9. $n \leqq N$ であれば Step2. へ，そうでなければ終了

5. ま と め

それぞれのディスクの性能・容量が等しい環境において，プライマリデータとバックアップデータのアクセス頻度の違いを利用して，各ディスクのアクセス量とデータ量を均衡化させる手法である AOD(Adaptive Overlapped Declustering) 法を拡張し，それぞれのディスクの性能・容量が違う環境においてもシステム容量を最大化し，BSR(Bandwidth to Space Ratio) を均衡化させる手法を示した．このとき，バックアップデータを左右に分けることによりディスク容量の差を吸収し利用可能容量を最大化し，隣接ディスク間でプライマリデータの割合を変えることにより性能の違いを吸収し，BSR の均衡化を行った．

本稿で示した手法はプライマリデータとバックアップデータを配置する領域を決定する手法である．実際のデータ量及び読み書き速度はデータの配置のされ方及びアクセスに依存する．例えば，各領域に均等に配置されず，一部分にのみ集中した場合は容量・性能の均衡化は果たせない．

今後の課題として，実システムに応用するためには，実際のデータ配置アルゴリズムや割り当て後のデータ再配置アルゴリズム (データマイグレーション) が必要である．

謝　辞

本研究の一部は，科学技術振興機構戦略的創造研究推進事業 CREST，文部科学省科学研究費補助金 n 特定領域研究 (19024028) および東京工業大学 21 世紀 COE プログラム「大

規模知識資源の体系化と活用基盤構築」の助成により行なわれた.

文　献

[1] Akitsugu Watanabe and Haruo Yokota. Adaptive overlapped declustering: A highly available data-placement method balancing access load and space utilization. pp. 828–839. 21st International Conference on Data Engineering (ICDE'05), 2005.

[2] Edward Grochowski and Roger F. Hoyt. Future trends in hard disk drives. In *IEEE TRANSACTIONS ON MAGNETICS*, Vol. 32, MAY 1996.

[3] Asit Dan and Dinkar Sitaram. An online video placement policy based on bandwidth to space ratio (bsr). *SIGMOD Rec.*, Vol. 24, No. 2, pp. 376–385, 1995.

[4] T. Cortes and J. Labarta. A case for heterogeneous disk arrays. *IEEE International Conference on Cluster ComputingIEEE International Conference on Cluster Computing (Cluster'00)*, p. 319, 2000.

[5] Yuewei Wang and David Hung-Chang Du. Weighted striping in multimedia servers. In *International Conference on Multimedia Computing and Systems*, pp. 102–109, 1997.

[6] T. Cortes and J. Labarta. Extending heterogeneity to RAID level 5. In *Proceedings of the 2001 USENIX Technical Conference*, pp. 119–132, Boston, 2001. USENIX Association.

[7] Roger Zimmermann and Shahram Ghandeharizadeh. Continuous display using heterogeneous disk-subsystems. In *ACM Multimedia*, pp. 227–238, 1997.

[8] Jen-Wen Ding and Yueh-Min Huang. Resource-based striping: An efficient striping strategy for video servers using heterogeneous disk-subsystems. *Multimedia Tools and Applications*, Vol. 19, No. 1, pp. 29–51, 1 2003.

[9] D.J. Hsiao, H.I .and DeWitt. Chained declustering: a new availability strategy for multiprocessor database machines. In Data Engineering, editor, *Proceedings. Sixth International Conference on*, 1990.

付　録

1. L_n, R_n を正しく割り当てるための条件

L_n, R_n に正しく 0 以上の値が割り当てられるために必要な条件について考察する．まず，バックアップ作成可能な領域の容量として，$D_n \equiv C_n - G_n (= L_n + R_n)$ とする．

次に，ディスク n から右に m 台目のディスクの領域 L_{n+m} は式 (A·1) のようになる．

$$L_{n+m} = \sum_{k=1}^{m} (-1)^{k-1} D_{n+m-k} + (-1)^m L_n \qquad \text{(A·1)}$$

条件より全ての m について $L_{n+m} \geqq 0$ が必要であるので，L_n の値は全ての m に対して，式 (A·3) または式 (A·4) を満たす必要がある．なおディスク $n+m$ の L_{n+m} とディスク n の $(-1)^m L_n$ の差を δ_c とおき，m が偶数と奇数のときで分解した．

$$\delta_c(n,m) \equiv L_{n+m} - (-1)^m L_n = \sum_{k=1}^{m} (-1)^{k-1} D_{n+m-k} \qquad \text{(A·2)}$$

$$L_n \leqq \delta_c(n,m) \qquad (m \in M_{odd}) \qquad \text{(A·3)}$$

$$L_n \geq -\delta_c(n,m) \qquad (m \in M_{even}) \qquad \text{(A·4)}$$

全ての n, m に対してこの式 (A·3) 式 (A·4) を成り立たせる L_n が存在するためには，式 (A·5) が成り立つ必要がある．

$$max_{x \in M_{even}}(-\delta_c(n,x)) \leqq min_{y \in M_{odd}}(\delta_c(n,y)) \qquad \text{(A·5)}$$

つまり，L_n, R_n に正しく 0 以上の値を割り当てるためには条件として式 (A·5) が成り立つことが必要である．

2. n によらず常に成立することの証明

式 (A·5) は n によらずに常に成立することを証明する．まず $\delta_c(n+1, m-1)$ と $\delta_c(n,m)$ の関係は式 (A·6) のようになる．

$$\begin{aligned}\delta_c(n+1,m-1) &= \sum_{k=1}^{m} (-1)^{k-1} D_{n+m-k} - (-1)^{m-1} D_n \\ &= \delta_c(n,m) - (-1)^{m-1} D_n \qquad \text{(A·6)}\end{aligned}$$

この式についても m が奇数のときと m が偶数のときで分けると，式 (A·7) と式 (A·8) とに分解できる．

$$-\delta_c(n+1,m-1) = -\delta_c(n,m) + D_n (m \in M_{odd}) \qquad \text{(A·7)}$$

$$\delta_c(n+1,m-1) = -(-\delta_c(n,m)) + D_n (m \in M_{even}) \qquad \text{(A·8)}$$

ここで，m が奇数のときは式 (A·7) が成り立ち，$-\delta_c(n+1, m-1)$ は $\delta_c(n,m)$ を正負反転し，D_n プラスした値となることを意味する．同様に，m が偶数のときは式 (A·8) が成り立ち，$\delta_c(n+1, m-1)$ は $-\delta_c(n,m)$ を正負反転し，D_n プラスした値となることを意味する．これより，m が奇数のときも m が偶数のときも $m \to m+1$ に変化させる際に同じ計算が行われる．そのため，ある n について式 (A·6) が成り立つとき，必ず $n+1$ についても式 (A·6) は成り立つ．よって，式 (A·6) は n によらず常に成立する．

なお n によらないので，以降 $n = 1$ とし $\delta_c(m) = \delta_c(1,m)$ と定義しなおす．

3. 境界条件

次に $m = N$ のときの条件 (境界条件) について考える．このとき $L_{1+N} = L_1$ であり式 (A·9) が成り立たなければならない．

$$L_1 = L_{n+N} = \delta_c(N) + (-1)^N L_1 \qquad \text{(A·9)}$$

この式を N が奇数のときと N が偶数のときで分けると，次の式 (A·10) 及び式 (A·11) に分解できる．

$$\delta_c(N) - 2L_1 = 0 \quad (N \text{ is odd}) \qquad \text{(A·10)}$$

$$\delta_c(N) = 0 \qquad (N \text{ is even}) \qquad \text{(A·11)}$$

つまり，$m = N$ の時点において式 (A·10) と式 (A·11) が成り立たないといけない．

4. G_n の変更による $\delta_c(m)$ の変化

G_n を変化させたときの $\delta_c(m)$ の変化について考察する．まずディスク l において $G_n = f$ とした時の $\delta_c(m)$ の値を $\delta'_c(m)$ とすると，次の式 (A·12) のようになる．

$$\delta'_c(m) = \begin{cases} \delta_c(m) & (m \in [1..l-1]) \\ \delta_c(m) - f & (m \geqq l \text{ and } m \in M_{odd}) \\ \delta_c(m) + f & (m \geqq l \text{ and } m \in M_{even}) \end{cases} \qquad \text{(A·12)}$$

ここで，l が奇数のとき，$l+n$ は n が偶数のとき奇数であり $-\delta'_c(m) = -\delta_c(m) + f$ となり，n が奇数のときは偶数であり $\delta'_c(m) = \delta_c(m) + f$ となる．よって l 番以降は全て $+f$ されることになる．l が偶数のときについても同様である．

結局 l 番目以降全て同じ値変更されるため，$l-1$ 番目と l 番目の大小関係以外には影響はない．