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

2007－DBS－143（72）
2007／7／3

# 二次記憶における多次元データのコンテナ化方式

嶋田 鉄兵† 都司 達夫† 樋口 健†

† 福井大学大学院工学研究科 〒910-8507 福井県福井市文京 3-9-1
E-mail: †{shimada,tsuji,higuchi}@pear.fuis.fukui-u.ac.jp

**あらまし** 多次元データベース等で扱われる多次元データを格納する多次元配列は通常，疎配列になる．またこの多次元配列は，集約演算や検索において，アクセス速度が配列要素のアクセスされた次元に依存するという次元依存性の問題がある．本論文では，チャンクの概念を導入したZ順序等の空間充填曲線に基づいて，これらの問題を解消すると同時に高い圧縮効果を実現する多次元データのコンテナ化アルゴリズムを提案し，評価する．

**キーワード** MOLAP，チャンク，コンテナ，Z順序化

# Containerization Scheme for Multidimensional Data on Secondary Storage

Teppei SHIMADA†, Tatsuo TSUJI†, and Ken HIGUCHI†

† Graduate School of Engineering, University of Fukui Bunkyo 3-9-1, Fukui-shi, Fukui, 910-8507 Japan
E-mail: †{shimada,tsuji,higuchi}@pear.fuis.fukui-u.ac.jp

**Abstract** Multidimensional arrays to store multidimensional data used in multidimensional database are often sparse. They also suffer from the problem that the time consumed in sequential access to array elements heavily depends on the dimension along which the elements are accessed. In this paper, against these problems we propose some containerization algorithms using space-filling curves such as Z-ordering, by introducing the notion of the *chunk*. We evaluate about the performance of our algorithms.

**Key words** MOLAP, chunk, container, Z-ordering

## 1. はじめに

近年，企業や組織において，大規模データベースに格納された大量データを分析し，意思決定支援に利用することが注目されている．これを行うためのシステムがOLAP（On-Line Analytical Processing）システムであり，ROLAP（Relational OLAP）とMOLAP（Multidimensional OLAP）に分類される．このうちMOLAPは，フロントエンドの関係データベースから定期的（例えば週ごと，月ごと）にデータを多次元配列にダンプし，バックエンドの多次元データベースとして高速に分析処理を行う．しかし，MOLAPで扱われる多次元配列は，以下の2つの問題を持つ．

（1） 配列が疎になる（疎配列）．

（2） 検索や演算などで配列へアクセスするのに費やされる時間が，アクセスする次元に強く依存する（次元依存性）．

（2）の問題は，配列をメモリに割り当てる方法に起因する．配列領域は通常，予め固定された次元順に線形に配置される．この場合，その次元順に沿うシーケンシャルアクセスは速いが，それ以外のアクセスは遅くなる，という問題が起きる．例えば，配列のスライス[6]に対する集合演算の応答時間は，スライスした次元に依存して大きく異なる．

次元依存性を解消しすべての次元に対して一様な扱いをするため，よく用いられる方法として，[1]で提案されているように，配列をチャンクと呼ばれる一様な大きさの部分配列に分割し，管理する方法がある．チャンク化では，n次元配列をそれより小さなサイズのn次元チャンクに分割し，二次記憶上で近接しているページとして各チャンクを格納する．配列のチャンク化により，前述の次元依存性は緩和できる．

チャンクを単位として二次記憶上のデータ管理を行う研究は盛んに行われているが[3][4][5]，予め静的に決められた次元順にディスクページに詰め合わせれば新たな次元依存性を生じうる．本研究では，チャンクコンテナの概念を導入し，空間充填曲線に基づいて，圧縮したチャンクをコンテナに詰め合わせ，二次記憶上のページに配置するためのコンテナ化アルゴリズムを提案するとともに，その効率を評価する．

## 2. 疎配列問題と次元依存性

関係テーブルを多次元配列にダンプする場合，MOLAPでは

配列の各添え字にカラムを対応付け，レコードと対応する位置の配列要素に集計値などのファクトデータを格納する．しかしこの方法で配列を構成した場合，各属性の値のすべての組み合わせに対してデータが存在する場合（密な状態）は極めて少なく，通常は多次元配列上のほとんどの要素が空の状態（疎配列）となる．このような疎配列問題は，配列の次元数（カラム数）と次元長（各属性の値の数）が増すほど深刻になる傾向にある．

一方，配列領域を確保する場合，メモリには配列のアドレス関数により線形に確保される．これは，多次元配列を二次記憶上に形成するMOLAPにおいても多く採用されている．この時，配列要素を格納した次元方向に沿う検索で十分なアクセス速度があったとしても，それに沿わない検索においてはアクセス速度が極端に低下する，という次元依存性の問題が存在する．

例として図1（A）のような各次元長8の3次元配列を考える．この例で，配列要素はX軸に沿って配置されているとし，二次記憶上の1ページには配列要素が8個入るものと仮定する．(A) において，X-Z平面のスライス検索では8回のページ読み込みが必要である．一方，Y-Z平面のスライス検索では64回，つまりすべてのページに対してアクセスする必要がある．

MOLAPにおいて，このような次元依存性の問題は重大である．OLAPの目的は対話的な多次元分析環境の提供にあり，ユーザのどのような検索要求に対しても応答時間が一定であることが望ましい．特にスライス処理のようなユーザの分析要求は，予め特定することができず，分析視点によって応答に差が生じることは好ましいことでない．

## 3. チャンク圧縮とコンテナ化

本章では，前章で示した疎配列問題と次元依存性を解決する方法について述べる．

### 3.1 チャンクとチャンク圧縮

チャンクとは，多次元配列の部分配列であり，n次元配列のチャンクはまたn次元配列である．チャンクのサイズは二次記憶上の1ページ以内であり，チャンクの要素は連続したディスク記憶を占める．ページは，主メモリと二次記憶の間のデータ転送の単位であるから，チャンクは1回の二次記憶からのページフェッチによって主メモリ上に読み込むことができる．チャンク化を行うことで，スライス面の取り出しに必要なディスクアクセス回数が，スライスする方向に関係なく平均化され，次元依存性を抑えることができる．

例として，図1（B）のような各次元長8の3次元配列を考える．この図において，配列を各次元長2のチャンク（8つの配列要素を含める）に分割する．この時，どの次元方向に対するスライス処理についても，ページ読み込み回数が16回となり，チャンクの導入によって次元依存性の問題を抑制できる．

チャンクによるページアクセスの平均化は，多次元配列が密な場合には効果的である．しかし，疎配列についてはチャンク化しても有効データのない記憶容量が大部分を占めるため，チャンクを圧縮する必要がある．すなわち，有効な配列要素のみ二次記憶に格納することにより，記憶容量の低減が可能となる．データを圧縮する方法としては，チャンク内の有効データをもつ配列要素の位置（オフセット）とデータを組として保持するチャンクオフセット圧縮[5]や，bitmapを保持するbitmap圧縮[6]などがある．各圧縮方式におけるデータ密度と記憶サイズの関係については[6]で述べられているが，本論文においてはこの圧縮方法は問わない．

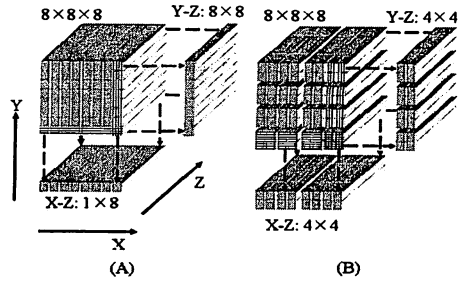


図1 8×8×8配列のチャンク化

### 3.2 コンテナ化

チャンクが疎な場合，圧縮されたチャンクのサイズは二次記憶上の1ページ以内に抑えられ，圧縮チャンクは1ページ以内に複数まとめて配置する事が可能である．これにより，1回のページ読み込みで複数のチャンクを読み込むことができる．このように，圧縮したチャンクを複数まとめて格納するための二次記憶上の1ページをチャンクコンテナ，または単にコンテナと呼ぶ．チャンクは論理的な単位である一方，コンテナは二次記憶の物理的な単位であることに注意されたい．ただし，チャンクはコンテナに線形に格納されるため，コンテナ化におけるチャンクの選択方法によっては，新たな次元依存性の問題が発生する．例えば，通常の配列要素の配置のように特定次元の順序で詰めた場合，次元順について新たな次元依存性が発生する．この問題を回避するには，コンテナを構成する際に，特定の方向性を持たないようチャンクを選択する必要がある．

コンテナの充填率を100%に近づけると，全体のコンテナ数は減少し，検索時のコンテナの読み込み回数が減少する．また，OLAPにおけるスライス処理では隣接したチャンク内の配列要素を必要とする．よってコンテナ化においては，隣接するチャンクをなるべく同一コンテナに格納し，かつ各コンテナの充填率を100%に近い高充填率にすることが望ましい．

## 4. コンテナ化の方式

本章では，3.2節で述べた問題点を考慮した上で，チャンクを選択しコンテナに格納するためのいくつかのコンテナ化アルゴリズムを提案する．以下提案するコンテナ化アルゴリズムにおいては，いずれもコンテナバッファとして1ページ分の主メモリ上の記憶領域を確保し使用する．コンテナ化では，アルゴリズムにしたがってチャンクを選択し，それをまずコンテナバッファに格納していく．コンテナバッファにそれ以上チャンクが格納できなければ，バッファの内容を1つのコンテナとして確定し，二次記憶上のコンテナファイルに書き込む．こうした一連の操作を，多次元配列のすべてのチャンクがコンテナ

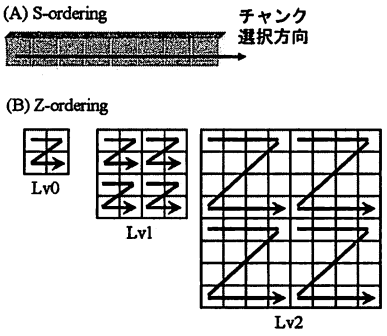


図 2　コンテナ化の方式

ファイルに書き込まれるまで繰り返す．図 2 に，提案するコンテナ化方式のチャンク選択順序の略図を示す．

### 4.1 Static Order

Static Order (S-order) は，通常の配列のように，チャンクを予め固定された順序について線形にコンテナバッファに格納する．コンテナバッファにチャンクがこれ以上格納できないならば，バッファを二次記憶上のコンテナファイルに出力する．この処理を，すべてのチャンクに対して繰り返し行う．ただし，この方法は次元依存性の解消を考慮しておらず，本論文ではその他のコンテナ化アルゴリズムとの比較対象とする．

### 4.2 Z 順序化

Z 順序は多次元データの任意の次元方向の近傍データを線形に配置する順序化の方式であり，Z 曲線を用いて多次元配列の各要素を "Z" の順序に順序付ける．本研究では，Z 順序をチャンク選択に用いる．コンテナ化においては，図 2 (B) に示すように，チャンクを Z 曲線によって選択し，順次コンテナバッファに格納していく．バッファにそれ以上チャンクを格納できないならば，S-order と同様，二次記憶上のコンテナファイルに出力していく．

Z 順序の特徴は，Z 曲線による近傍性の確保，方向性の低さが挙げられる．これらの特徴は次元依存性の解消につながり，スライス検索が頻繁に行われる MOLAP では有利に働くことが期待できる．しかし，コンテナの充填率を優先した場合，Z 曲線によって離れた部分のチャンクを同一コンテナに格納する可能性があり，コンテナ読み込み回数の増加につながる．この問題を解決する方法として，Z 順序化における再帰のレベルに応じた分割方式を提案する．

### 4.3 Z 順序のレベル分割

Z 順序において，Z 曲線による「レベル」の規定を行う．図 2 (B) に示すように，最小の Z 曲線で指定できる領域をレベル 0，その 2 倍の領域をレベル 1，4 倍の領域をレベル 2，… と呼ぶ．また，レベルにより分割される各領域をそのレベルのレベル領域という．Z 順序を用いたコンテナ化においては，レベルをコンテナ化の単位とする．つまり，事前に指定したレベルに従い，レベル領域内のチャンクを Z 順序で指定し，コンテナバッファに格納していく．1 つのレベル領域について処理が終了したなら，その時点で満杯でなくてもバッファをコンテナファイルに出力し，次のレベル領域ではチャンクを新たなコンテナバッファに格納していく．

レベル分割で注意すべき点は，指定するレベルによってコンテナ数が変動する点である．レベルが小さい場合は各次元方向でのコンテナ数が一定となり易く，コンテナの読み込み回数が一定となる．しかし低充填率のコンテナが大量に発生し，コンテナ数が大幅に増加する恐れがある．一方，レベルが大きい場合は低充填率のコンテナが発生する割合は減る．だが，各次元方向でのコンテナ数がばらつくため，コンテナの読み込み回数は各次元で変動することになる．よって，Z 順序のレベル分割を適用する場合は，多次元配列の充填率から最適なレベルを指定し，コンテナ化する必要がある．

## 5. コンテナ集合の効率性を重視した Z 順序化の方式

### 5.1 提案方式の要点

前章で提案した Z 順序化およびそのレベル分割は，特に近傍性の面で他のアルゴリズムより優れている．しかし，Z 順序のコンテナ化は 2 のべき乗の領域で規定されており，これに沿わない一般の多次元配列では特定次元への次元依存性が発生する可能性がある．また Z 順序のレベル分割では，高レベルほど各レベル領域のチャンク数が増え，レベル領域について複数のコンテナが生成されるという事が起こる．これに加え，端部分にわずかなチャンクが残り，低充填率のコンテナが生成されるという問題もある．これはコンテナ数の増加につながり，よってアクセス時のコンテナ読み込み回数を増加させる要因となる．

以降の節では，これら Z 順序がもつ問題点を解消するための方法を述べる．これらの方法では，Z 順序のもつ近傍性の特性を保つ一方，コンテナ化で形成するコンテナの充填率を上げることの双方を考慮している．

### 5.2 相似形分割

Z 順序化では，そのコンテナ化範囲を 2 のべき乗の大きさで規定している．しかし，フォアグラウンドの OLTP データベース上の関係テーブルにおいては，各カラムのカージナリティは同一ではなく大きく異なることもある．したがって，バックグラウンドの多次元配列にダンプした場合に，配列の各次元長は等しくなく，2 のべき乗でもない場合がほとんどである．よって，スライス次元により断面上の要素数は大きく異なり，その検索時間にも大きなばらつきを生じ得る．チャンクをコンテナに格納する順序に起因する次元依存性とは別に，このようなカージナリティのばらつきにより生じる検索時間の次元依存性を同時に解消することが望ましい．

この次元依存性を緩和する方法として，ここでは全体のチャンク配列に相似なチャンク集合をコンテナ化の単位とする方式を提案する．相似形は，元の多次元配列を適当な倍率で各辺の比を保つよう縮小した形状をもつ．さらにこの部分配列のサイズを二次記憶上の 1 ページ以内に設定することで，コンテナ化の単位とすることが可能である．例えば，図 3 のような 6 × 4 × 5 のチャンク配列においては，3 × 2 × 2 の相似形に分割でき，それぞれをコンテナ化の単位とすることができる．

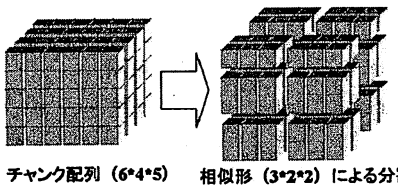


図 3　相似形分割

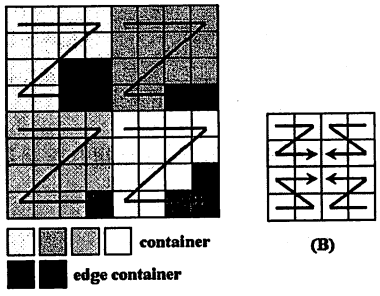


図 4　端部分の集積方式

ただし，相似形が極端に小さくなる場合は，それぞれの相似形で Z 曲線が描けず，S-order に似たチャンク選択を行う恐れがある．そこで，相似形の範囲が通常の Z 曲線の範囲より小さくないか，またその範囲内の充填率が 100% 未満かを調べ，可能ならば各辺を 2 倍に拡大し新たな相似形（調整形）をつくる．

### 5.3　充填率と近傍性を考慮した Z 順序化のコンテナ方式

4.3 節において Z 順序のレベル分割を述べたが，この方法についても問題がある．分割レベルが大きい場合，領域内のチャンクの数が多くなるため，1 つのコンテナで領域内すべてのチャンクを格納することができず，領域端部に低充填率のコンテナが形成される可能性がある．これはコンテナ数の増加につながり，検索時のコンテナ読み込み回数を増加させる要因となる．

この問題を解消する方法として，Z 順序の各レベル領域の端部に残ったチャンクを集積し，コンテナとしてまとめる方法を提案する．本節では，その方法として 2 種類の方法を提案する．

#### 5.3.1　端部分の集積方式

前述の問題を解消する最も単純な方法は，Z 順序の各レベル領域の端部分に残ったチャンクを集積し，コンテナとしてまとめる方法である（図 4 (A)）．この方法では，Z 順序のレベル分割において，各レベル領域で端部分に残ったチャンクを，Z 順序で近傍性を考慮しながら別のコンテナバッファに格納していく．また通常のレベル分割とは異なり，端部分用のバッファについては，充填率を優先して，指定レベルを越えて他のレベル領域にある端部分のチャンクをコンテナ化していく．これによって，Z 順序のもつ近傍性を保持しつつ，コンテナ数の低減化，各コンテナの高充填率化を図ることができる．

#### 5.3.2　周回型 Z 順序

周回型 Z 順序とは，Z 順序の方向を変えることで端部分となるチャンクを中心に寄せ，コンテナ化する方式である（図 4 (B)）．コンテナ化処理では，各レベルにおいて Z 曲線の方向を決め，それに従いチャンクを指定，コンテナ化していく．ここで Z 曲線の方向は，多次元配列の中心に Z の方向が向かうように決める．中心に寄せられた端部分のチャンクは，前節で述べた集積方式と同様に，通常のコンテナバッファとは別のコンテナバッファに順次詰めていき，最後に両者を合わせて最終のコンテナを得る．中心部分に端部分を寄せることにより，端部分のチャンクを，近傍性を考慮してコンテナ化することができる．また，Z の方向を回転させることで，Z 曲線の方向の規則性をある程度なくし，各次元でのコンテナ読込回数を平均化させる効果があると考えられる．

## 6.　評　　価

第 4. 章，第 5. 章で提案したコンテナ化アルゴリズムについて，その有効性をシミュレーションにより評価する．

### 6.1　評価条件

ここでは，各方式の特性を見るため，それぞれについて評価する条件を指定する．

（1）　コンテナ化アルゴリズムの性能比較
（2）　相似形による性能の評価
（3）　Z 順序における端部分の処理での比較

（1）については，1 つの次元に 32 個のチャンクをもつ 5 次元チャンク配列について，すべてのチャンクが一様なデータ密度（0.01% ～0.1% ）をもつ場合について，チャンクの充填率を 0.01% 刻みで変化させて評価を行った．

（2）については，各次元がそれぞれ 32，16，32，64，32 個のチャンクをもつ 5 次元チャンク配列について，チャンク配列が一様分布（0.01% ～0.1% ）をもつ場合について，各次元での評価を行った．また，相似形の大きさは各次元それぞれ 2,1,2,3,2 とし，その各次元長を 2 倍した調整形を用いた．

（3）については，各次元がそれぞれ 32，16，32，64，32 個のチャンクをもつ 5 次元チャンク配列について，すべてのチャンクが一様なデータ密度（0.1% ）をもつ場合について，各次元での評価を行った．

### 6.2　実験内容

シミュレーション実験においては，コンテナ総数，コンテナの平均読み込み回数，および 1 回の平均読み込み回数に対する標準偏差を計測する．

ここではチャンク総数を固定するので，コンテナ総数はコンテナへの圧縮の度合いを示す．他の 2 つについては，4 次元スライスに対する測定を行う．コンテナの平均読込回数は，4 次元断面上にあるコンテナ数の平均である．平均が高いほど読み込みが必要なコンテナが増え，検索時間が増加する．スライス断面上のコンテナ数の標準偏差は，スライス検索時間のばらつきを表す．値が低いほどばらつきが低いといえ，次元依存性の程度を示す指標となる．

また 6.1 （3）については，範囲検索を用いての測定を行う．ここでは，指定した次元に 3 チャンク分の幅を，それ以外の次元ではすべての範囲を対象として，その範囲を指定した次元について 1 チャンクずつずらしてコンテナ読み込み回数をそれぞ

れし，その平均および標準偏差を算出して評価に用いる．

### 6.3 実験結果および考察

#### 6.3.1 コンテナ化アルゴリズムの性能比較

図5は，チャンク充填率を変化させたときの，コンテナ化アルゴリズム別のコンテナ総数の推移を示す．図において，S-orderとZ順序が最も低い推移を示す．これは，両方式とも充填率を優先させて順次チャンクをコンテナに格納するためである．一方，Z順序のレベル分割の結果を見ると，レベル別に結果が大きく異なることが分かる．レベル分割方式のうち，レベル1は実験の充填率においてほとんど一定の値を示している．これはレベル1の範囲が小さく，充填率の低いコンテナがレベル領域ごとに生成されるためである．ただしチャンク充填率が0.1%の場合では，各領域の充填率が100%を超えるため，各領域で複数のコンテナが生成され，よってコンテナ総数が2倍に急増した．レベル2についてはレベル領域の範囲が大きいため，Z順序に漸近する形で結果が推移する．レベル2は高充填率のコンテナが生成されやすい一方，各領域の端に低充填率のコンテナが形成されるため，Z順序よりも平均4.8%高くなっている．

図6は，4次元スライスをとった時のコンテナの平均読み込み回数を示す．コンテナ総数で低い推移をとるZ順序とレベル2の結果が低く，コンテナ総数の多いレベル1が高い推移を示す．S-orderは一方向にのみ連続したコンテナ化を行うため，コンテナ化方向以外のスライス処理に対し多くの読み込み回数を必要とし，よって一番高い推移を示す．

図7は，図6の結果に対する標準偏差を示す．結果はZ順序とS-orderで大きく差があり，S-orderはZ順序に対し約15倍の値となった．これは，前述と同様S-orderのアルゴリズムにより，各スライスで得られるコンテナ数に偏りが生じるためである．その他の結果をZ順序を基準として見ると，レベル2はZ順序に対し3.4〜(-0.056)%の間で漸近している．一方，レベル1は各レベル領域が小さいため各次元方向のコンテナ数が揃い，チャンク充填率0.1%の場合を除いて0となった．

以上の結果から，Z順序がコンテナ総数および平均読み込み回数で有利であることが分かり，レベル2がこれに漸近する．レベル1は標準偏差で優れているが，コンテナ総数が高い点で他に劣る．これらは，第4.章で述べたZ順序のもつ特性（近傍性の良さ，方向性の低さ）によるものと考えられ，Z順序が多次元配列のコンテナ化に適しているということがいえる．

#### 6.3.2 相似形

S-orderの結果は前節の結果と同様の傾向を示しているので，省略する．

図8は，アルゴリズム別のチャンク充填率に対するコンテナ総数の変化を示す．各アルゴリズムが似た傾向を示していることが分かる．図において，Z順序の相似形導入前後の結果は変化なく，最小である．一方，レベル分割方式においては，相似形の導入でレベル1，2ともコンテナ総数の減少が起こっている．レベル2は元の方式より平均4.1%，レベル1においては47%低くなっている．これは相似形（を調整したもの）をレベルの単位としたことで，各レベル領域内のコンテナ充填率が上がったためと考えられる．これらは元の方式よりもZ順序の結

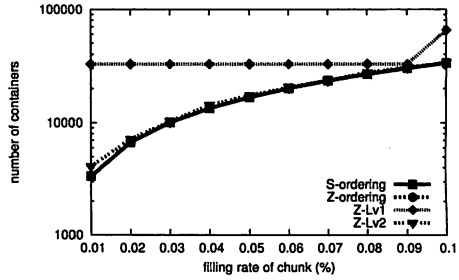


図5 コンテナ総数

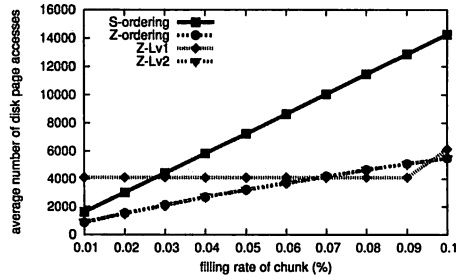


図6 コンテナ平均読込回数

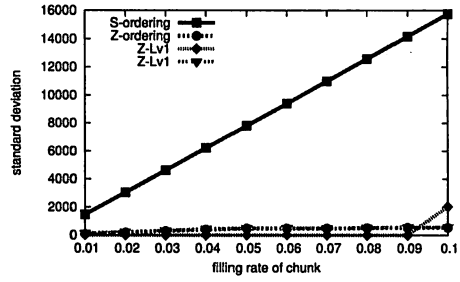


図7 コンテナ平均読込回数に対する標準偏差

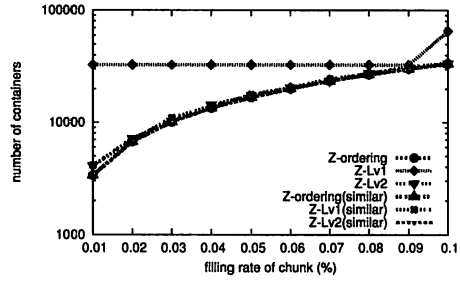


図8 コンテナ総数（相似形）

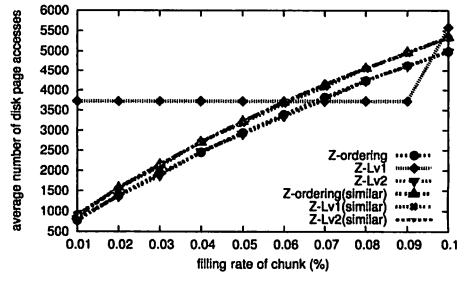


図9 コンテナ平均読込回数（相似形）

果（最小コンテナ総数）により漸近しており，レベル1は平均5.2%，レベル2は0.19%，Z順序よりも高い．

図9は，4次元スライスに対するチャンク充填率別のコンテナ平均読み込み回数を示す．一見して，コンテナ総数とほぼ同様の傾向で結果が推移しているように見える．しかし，元の方式と比べて，相似形を用いたZ順序は平均して8.9%，レベル2は8.8%値が高くなっている．これは，元々高充填率のコンテナを生成していた方式に相似形を導入したことで，より多くの高充填率コンテナが生成される一方，各次元方向のコンテナがよりいびつになったことによると考えられる．一方，相似形を用いたレベル1は元の方式よりも平均16.0%，最大76%値が低くなっている．これは元の方式のレベル領域が小さかったため，相似形の導入で高充填率のコンテナが生成でき，コンテナ総数が激減したことによると考えられる．

図10は，図9の結果に対する標準偏差を示す．図より，いずれの方式についても相似形の方が元の方式より明らかに低い結果となった．これは，相似形を使用したことで，各次元方向のコンテナ数が均一に揃いやすくなり，各次元方向でのコンテナ数のばらつきが解消されたことによると考えられる．

まとめると，相似形分割は各次元方向のコンテナ数を均一にする効果があり，特に標準偏差において大幅な改善がみられた．

### 6.3.3 端部処理の有無

レベル分割を行わないZ順序については端部処理を行っても結果が変わらないため，対象から除外する．

図11は，チャンク充填率が0.1%一様分布のときの，各アルゴリズムに対する次元別のコンテナ平均読み込み回数を示す．端部分を集積するレベル分割方式の方が，元の方法に比べて値が小さいことが分かる．また，集積方式別では周回型の方が端部集積型よりもレベル1，2ともに平均0.24%値が低い．

図12は，図11に対する標準偏差を示す．この結果でも，相似形に対しわずかな値の減少が見られ，レベル1で最大0.3%，レベル2で最大5.1%の減少となっている．レベル1の減少が少ないのは，元々レベル1のレベル領域が小さく，端部分のチャンク数が少ないために，集積によるコンテナの減少効果が低かったからであると考えられる．また，集積方式別では周回型の方が端部集積型よりもレベル1，2ともに平均1.2%低い．

まとめると，端部分の集積方式はZ順序のレベル分割において，コンテナ総数の減少と，各次元方向のコンテナ数の差のを抑制する効果があり，周回型方式の方がより良いと考えられる．

## 7. むすび

本論文では，Z順序に基づいた多次元配列の圧縮と格納のためのコンテナ化方式を提案した．Z順序によるコンテナ化は，チャンクのコンテナへの圧縮度が高く，また近傍性の良さと方向性の低さから，各次元での読込回数ならびにその標準偏差についても良好な結果を示した．さらに，Z順序での相似形分割は各次元方向のコンテナ数を均一にし，標準偏差の点で元のZ順序より大幅に動作が改善されることが示された．このことはスライス処理の多いOLAPで非常に有利であると考えられる．

課題としては，実際のシステム上でのコンテナ化にこれらの手法を適用した場合の実行時間の測定が必要になると思われる．また，分布が不均一な場合や高いチャンク充填率における，これらの手法の有効性を測定する必要もある．

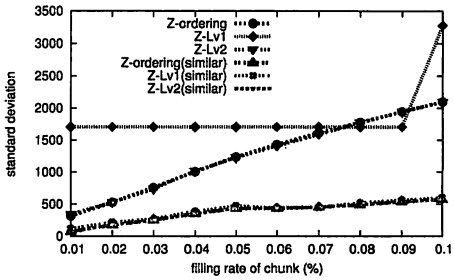


図10 コンテナ平均読込回数に対する標準偏差（相似形）

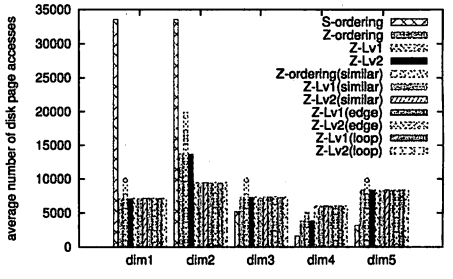


図11 次元別のコンテナ平均読込回数（端部処理）

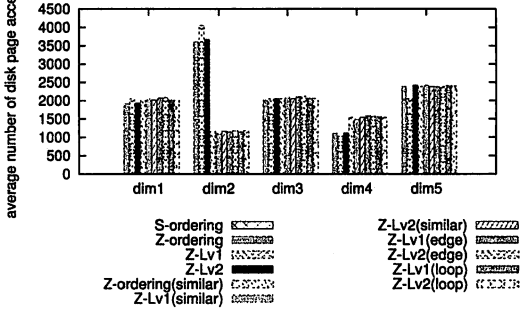


図12 次元別のコンテナ平均読込回数に対する標準偏差（端部処理）

## 文献


[1] S.Sarawagi, M.Stonebraker, Efficient Organization of Large Multidimensional Arrays, Proc. of ICDE, 1994, pp.328-336.

[2] Kent E. Seamons, Marianne Winslett, Physical Schemas for Large Multidimensional Arrays in Scientific Computing Applications, Proc. of SS-DBM, 1994, pp.218-227.

[3] P.M.Deshpande, K.Ramasamy, A.Shukla, and J.F.Naughton, Caching multidimensional queries using chunks, ACM SIGMOD, Seattle, WA, 1998, pp.259-270.

[4] Sanjay Goil, Alok Choudhary, A Parallel Scalable Infrastructure for OLAP and Data Mining, International Data Engineering and Applications Symposium, AZ, 1999, pp.159-170.

[5] Yihong Zhao, Prasad M.Deshpande, Jeffry F.Naughton, An Array-Based Algorithm for Simultaneous Multidimensional Aggregates, Proc. of SS-DBM, 1994, pp.218-227.

[6] Tatsuo Tsuji, Atsuo Isshiki, Teruhisa Hochin, Ken Higuchi, An Implementation Scheme of Multidimensional Arrays for MOLAP, DEXA Workshops 2002, pp.773-778.