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

2007－DBS－143（70）
2007／7／3

垂直分割に基づいた超高次元データからの並列頻出パターン発見手法

森 紘一郎† 折原 良平†

† 株式会社 東芝 研究開発センター 〒212-8582 神奈川県川崎市幸区小向東芝町1
E-mail: †{kouichirou1.mori,ryohei.orihara}@toshiba.co.jp

あらまし 従来の並列頻出パターン発見手法は，レコード数が非常に大きなデータに対し，データを水平分割して各計算機に割り当てる手法が一般的であった．しかし，近年，属性数が非常に大きな超高次元データからの頻出パターン発見が重要になってきた．このようなデータには，従来の水平分割に基づいた並列化手法では対応できない．本論文では，データの垂直分割とレコード空間探索を組合せた並列アルゴリズムが超高次元データの頻出パターン発見に有効であることを示す．提案手法を現実のマイクロアレイデータセットを用いて評価したところ16台で約13倍の速度向上が達成できた．

キーワード データマイニング，相関ルール，頻出パターン，高次元データ，垂直分割，並列計算

Parallel Frequent Pattern Mining Method from Super-High-Dimensional Data by Vertical Partitioning

Kouichirou MORI† and Ryohei ORIHARA†

† Research & Development Center, Toshiba Corporation
1 Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, Kanagawa, 212-8582 Japan
E-mail: †{kouichirou1.mori,ryohei.orihara}@toshiba.co.jp

Abstract In general, traditional parallel frequent pattern mining methods were applied to data that contains a large number of records. The data was horizontally partitioned and each partitioned data was allocated to processing elements. However recently, frequent pattern mining from super-high-dimensional data that contains a huge number of attributes is becoming important. The traditional parallel frequent pattern mining methods cannot handle these data. In this paper, we show that the combination of vertical partitioning and record space search is efficient for parallel frequent pattern mining of high-dimensional data. We evaluate our method with real microarray dataset on 16 PCs to discover that it is approximately 13 times faster than sequential one.

Key words Data Mining, Association Rule, Frequent Pattern, High Dimensional Data, Vertical Partitioning, Parallel Processing

1. はじめに

頻出パターン発見手法は，Apriori [1] をはじめとしてさまざまな手法が提案されてきた．従来手法の多くは，レコード数が極めて大きく，属性数が小さいデータを効率的に処理することに重点が置かれていた．極めて大規模なデータに対しては，並列計算が有効であり，さまざまな並列頻出パターン発見手法が提案されている [2] [3] [4] [5] [6]．従来の並列アルゴリズムの多くは2つの共通点を持つ．1つめは，データをレコード方向に分割して各計算機に配置するデータの水平分割を用いる点である．2つめは，頻出アイテム集合の探索において属性（アイテム）の組合せを探索する属性空間探索を用いる点である．この組合せは上述の特徴を持つデータには大変有効であることが示されている．

その一方で，近年，レコード数は小さいが，属性数が極めて大きいデータを対象とした分析への期待が高まっている．たとえば，マイクロアレイデータやセンサーデータなどである．このような属性数が極めて大きいデータでは，従来の属性の組合せを探索する手法では計算量が指数関数的に大きくなり実時間内での計算が困難という問題があった．この問題に対して，我々は，レコード数が小さく，属性数が極めて大きい超高次元データから頻出パターンを効率的に発見する逐次型アルゴリズムを提案している [7]．我々の逐次型アルゴリズムは，レコードに含まれるアイテム集合の共通集合演算を多用するため，属性数が大きくなると共通集合演算に多くの時間を費やすという問題点があった．そこで，本論文では，この問題を解決するため

新たな並列アルゴリズムを提案する．本手法は2つの特徴を持つ．1つめは，データを属性方向に分割して各計算機に配置するデータの垂直分割を用いる点である．2つめは，頻出アイテム集合の探索においてレコードの組合せを探索するレコード空間探索を用いる点である．本手法は，従来の並列頻出パターン発見手法とは，対象とするデータもデータ分割の方法もまったく逆の特性を持つ手法となっている．

2章では，従来の並列頻出パターン発見手法の概要を説明し，データの水平分割と属性空間探索が相性のよい組合せであることを示す．3章では，提案手法の逐次版と並列版アルゴリズムの詳細を説明する．4章では，提案手法を人工データセットとマイクロアレイデータセットに適用した結果について述べる．5章では，全体をまとめる．

2. 関連研究

頻出パターン発見は大規模なデータになるほど計算量が大きくなるためその並列計算に関する研究は数多くなされている．本章で述べるすべてのアルゴリズムは，分散メモリ型並列計算機（PCクラスタ）上で動作する．以後，分散メモリ型並列計算機を構成する各計算機をPE (Processing Element) と略記する．

Count Distribution (CD) [2] は，Apriori [1] ベースの並列アルゴリズムである．Apriori で多くの時間を費やす処理は，データベースをスキャンし，候補アイテム集合のサポートカウントを計算する部分である．CDはデータを水平分割して各PEに配置し，サポートカウントを並列に計算することで処理を高速化している．CDでは，各PEが同一の候補アイテム集合を格納するためメモリ効率が悪いという問題がある．

Hash Partitioned Apriori (HPA) [3] は，Apriori ベースの並列アルゴリズムである．HPAは，データを水平分割し，さらに候補アイテム集合を各ノードに分割することでCDのメモリ効率が悪いという問題点を解決している．

Parallel Data Mining (PDM) [4] は，基本的にCDと同様に各PEにデータを分割し，サポートカウントを並列に計算する．CDと異なるのは，最初のデータベーススキャン時にハッシュテーブルを作成してサポートカウントを近似的に計算し，候補アイテム集合の削減に利用する点である．

Partition [5] は，水平分割したデータからローカルサポートカウントを求め，その結果をマージしてグローバルサポートカウントを求める手法である．サポートカウントを求める際にアイテムとレコードを入れ替えた垂直データレイアウトを用い，2回のデータベーススキャンで頻出アイテム集合を得られるのが特徴である．

Fast Distributed Mining (FDM) [6] は，CDを改良した手法である．ローカル頻出アイテム集合を求めた時点でグローバル頻出でないと考えられるアイテム集合を候補から除外することで候補アイテム集合を削減し，他PE間の通信量を減らしている．

従来の並列頻出パターン発見手法には2つの共通点が見られる．1つめの共通点は，表1に示すようにレコード数が非常に大きく，属性数，平均レコード長が小さいデータを対象としている点である．これは，頻出アイテム集合発見手法がPOSデータを想定して開発されたアルゴリズムだからだと考えられる．そのため，上記の研究はすべて図1に示すようにデータをレコード方向に分割して各PEに配置する戦略を取っている．データをレコード方向に分割する方法は水平分割と呼ばれる．2つめは，頻出アイテム集合の探索において属性（アイテム）の組合せを探索する点である．これは，上記のアルゴリズムがAprioriを起源にしているためである．Apriori は，長さ k の頻出アイテム集合から長さ k+1 の頻出アイテム集合を発見するというように短い頻出パターンから長い頻出パターンへと属性を組合せて探索を進める．このような属性の組合せを探索する方法は属性空間探索と呼ばれる．

表1 テストデータセット

アルゴリズム	平均レコード長	レコード数	アイテムの種類
CD	5	3278K	記述なし
	10	2016K	記述なし
	15	1456K	記述なし
	20	1140K	記述なし
HPA	10	2048K	記述なし
	15	2048K	記述なし
	20	2048K	記述なし
PDM	10	250K	記述なし
	20	250K	記述なし
Partition	5	100K	1000
	10	100K	1000
	20	100K	1000
FDM	10	200K	1000

図1 データの水平分割

データの水平分割と属性空間探索は相性がよい組合せである．相性がよいとはデータを分割しても各PE間でプロセス間通信が頻繁に発生しないことを意味する．データを水平分割した場合，各PEは異なるレコードを割り当てられるが，割り当てられる属性はすべてのPEで共通であり，かつ元データのすべての属性が含まれている．これは，各PEで属性の組合せを生成する際に他のPEから属性情報を受け取らずにすべての属性組合せを生成できることを意味する．もしレコードではなく，属性が各PEに分割されてしまうと，各PEはプロセス間通信をせずにすべての属性の組合せを生成することはできない．

3. 提案手法

本章では，従来研究が対象としてきたデータと逆の特徴を持つデータ，すなわち，レコード数は少ないが，属性数が非常に大きな超高次元データから頻出パターンを効率的に発見する手

```
Input:
  D:database
  minsup:minimum support count
  minlen:minimum pattern length
Output:
  LFP: longest frequent patterns
  F: frequent patterns
Method:
  N[1] = gen_1-ridsets(D)
  for (k=2; k<=minsup and N[k]!=∅; k++) {
      C[k] = gen_ridset(N[k-1])
      N[k] = {n|len(n.itemset)>=minlen for n in C[k]}
  }
  F' = {n.itemset for n in N[minsup]}
  LFP = arg max_i len(i) for i in F'
  F = {s|s>=minlen for s in subset(F')}
  return F
```

図 2　逐次版の擬似コード

TID	アイテム
0	A B D E
1	B C E
2	A B D E
3	A B C E
4	A B C D E
5	B C D

図 3　サンプルデータベース

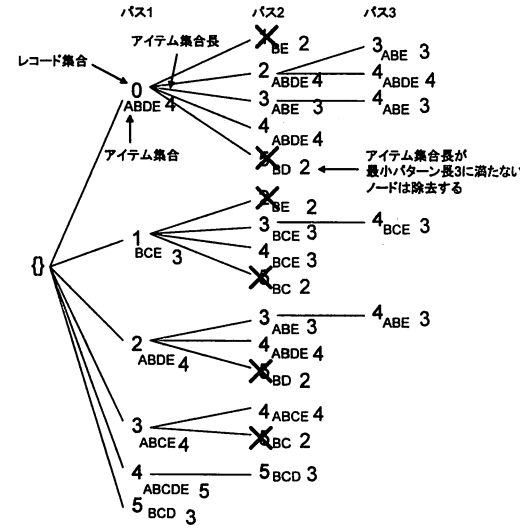

図 4　逐次版の処理例

法を提案する．逐次版アルゴリズムについては，[7] でアルゴリズムを詳説したので 3.1 節では，擬似アルゴリズムと例題を元に簡単に説明する．3.2 節では，逐次版アルゴリズム高速化のボトルネックが共通集合演算にあることを示し，データの垂直分割に基づいた並列計算手法を提案する．

3.1 逐 次 版

本手法が対象とするデータは，レコード数が少なく，属性数が非常に大きい超高次元データである．属性数が非常に大きいため属性の組合せを探索するアプローチでは組合せ爆発が生じ有効ではない．そこで，属性の組合せを探索するのではなく，レコードの組合せを探索し，レコードに含まれるアイテム集合に共通する属性を求めることで頻出パターンを発見する．レコード組合せを探索する手法は，レコード空間探索と呼ばれる．また本手法では，最小パターン長と呼ぶパラメータを導入し，発見されるパターンは最小パターン長以上の長さを持つ頻出パターンのみとした．最小パターン長の導入によってレコード空間探索で枝刈りを効率的に行えるようになり，パターン長が長い頻出パターンでも効率的に発見できる．

図 2 に提案手法の逐次版の擬似コードを示す．以下，本手法の特徴を図 3 のサンプルデータと図 4 の処理例を用いて説明する．アルゴリズムの詳しい手順と証明は，[7] を参照されたい．

図 3 は，図 1 中の関係データを変換したトランザクションデータベースである．一般的に関係データベースとトランザクションデータベースは相互に変換可能である．レコードはトランザクション，属性はアイテムに対応する．本論文では，レコードとトランザクション，属性とアイテムを同じ意味で用いる．

処理例では，最小サポートカウントを 3，最小パターン長を 3 と設定した．すなわち，サポートカウントが 3 以上で，かつパターン長が 3 以上の頻出パターンが抽出される．本手法では図 4 に示すようなトライと呼ばれる木構造を用いて探索を進める．Apriori の節点は通常アイテム集合であるのに対し，本手法では節点がレコード集合となるのが特徴である．図 4 には，木構造全体を図示しているが，実際はレコード集合長 1 から順に枝を伸ばして探索を進める．提案手法では，レコード集合とアイテム集合を対にして管理し，これをノード呼ぶ．たとえば，(0, ABDE)，(23, ABE) はノードである．

まず，長さ 1 のレコード集合である 0，1，2，3，4，5 を列挙し，各レコードに含まれるアイテム集合とアイテム集合長を求める．ここでは，最小パターン長 3 より短いアイテム集合はないため除去されるノードはない．

次に，長さ 1 のレコードから長さ 2 のレコード集合である 01，02，03，04，…，45 を生成する．このとき Join 操作によって生成したレコード集合のいずれかの部分集合がすでに除去されている場合は Apriori Property により生成しなくてよい．今回の例では Apriori Property によって除去できるノードはないが，たとえば，03 を生成したとき，03 の部分集合 0，3 のいずれかが前のパスですでに除去されていた場合，03 は生成する必要はない．

生成したレコード集合に対応するアイテム集合は個々のレコードに対応するアイテム集合の共通集合をとる．たとえば，レコード集合 0 に対応するアイテム集合が ABDE，レコード集合 1 に対応するアイテム集合が BE なのでレコード集合 01 に対応するアイテム集合は ABDE と BE の共通集合をとった BE となる．このように本手法では，アイテム集合の共通集合演算を頻繁に行う．この例題では，アイテム集合長が短いため共通集合演算のコストは小さいが，超高次元データになるとアイテム集合長が長くなるため共通集合演算のコストは大きくなる．

次に，長さ 2 のレコード集合に対応するアイテム集合長を求める．このとき，01 に対応するアイテム集合 BE の長さは 2 であり，最小パターン長 3 に満たないため対応するレコード集

合 01 はノードごと除去される．レコード集合 05，12，15，25，35 も同様の理由でノードごと除去される．

次に，長さ 2 のレコード集合から長さ 3 のレコード集合を生成する．このとき，Apriori と同様 Join 操作が可能である．たとえば，02，03，04 からは 023，024，034 が生成される．ここでも Apriori Property によって生成するかしないかを判断する．たとえば，034 を生成したとき，034 の部分集合 03，04，34 のいずれかのノードが前のパスですでに除去されていた場合，034 は生成する必要はない．ここで，034 の部分集合として 0，3，4 については除去されているか調べなくてよい．なぜなら，1 つ前のパスで 03 を生成するときに 0，3 が除去されているか，04 を生成するときに 0，4 が除去されているかすでに調べているためである．

023 に対応するアイテム集合は，02 と 03 の共通アイテム集合をとる．02 に対応するアイテム集合が ABDE，03 に対応するアイテム集合が ABE なので 023 に対応するアイテム集合は ABDE と ABE の共通集合である ABE となる．

この例では，最小サポートカウントが 3 であるためパス 3 で探索を打ち切る．最後に探索結果から最長頻出パターンと最小パターン長 3 以上の長さを持つすべての頻出パターンを生成する．最小サポートカウントが 3 であるためレコード集合長が 3 のレコード集合に対応するアイテム集合 F ' = {ABE, BCE, ABDE} を抽出する．F' に重複がある場合は除去される．このとき，F' に含まれるアイテム集合の最小サポートカウントは 3 以上であることが保証されている．たとえば，ABE について考えてみる．ABE に対応するレコード集合は 023 である．アイテム集合はレコード集合に対応するアイテム集合の共通集合をとってきたことから，レコード 0，2，3 に共通するアイテム集合が ABE といえる．つまり，ABE は少なくともレコード 0，2，3 に含まれており，ABE のサポートカウントは少なくとも 3 であることがわかる．

ここで，ABE のサポートカウントが 3 であるとは言い切れない．実際，ABE は，レコード 0，2，3，4 に含まれておりサポートカウントは 4 である(注1)．しかし，ここでは，アイテム集合が頻出であるか，頻出でないかわかればよいので ABE のサポートカウントが最小サポートカウントの 3 以上であることが保証されていればよい．

F' の中で最も長い ABDE が最長頻出パターンである．最小パターン長以上の長さを持つすべての頻出パターンは，長さが 3 以上の F' の全部分集合 F = {ABE, BCE, ABD, ADE, BDE, ABDE} である．

F の各アイテム集合の部分集合，たとえば，A，B，C，E，AB，BC なども頻出パターンではあるが，本手法では長さが最小パターン長より短い「すべての」頻出パターンが抽出できる保証はない．

3.2 並 列 版

逐次版アルゴリズムにおいて計算時間がかかる部分は，生成したレコード集合に対応するアイテム集合を計算するところである．このアイテム集合の計算では，2 つのアイテム集合の共通集合をとる．超高次元データの場合，アイテム集合長が長くなるため共通集合演算に時間がかかるという問題点がある．しかも共通集合演算はアルゴリズムの中でもっとも高頻度で生じる計算にあたる．そのため計算時間の大部分を占める共通集合演算部を並列計算できれば大きな速度向上が期待できると考えた．

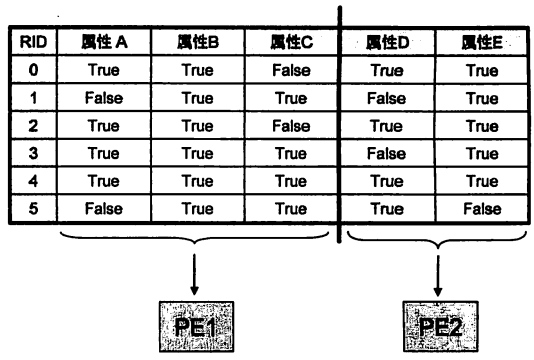

図 5　データの垂直分割

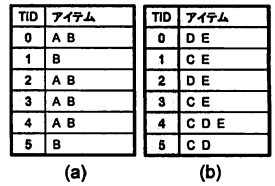

図 6　垂直分割したデータベース

この問題に対し，属性空間を分割して各 PE に配置するデータの垂直分割に基づく並列計算を用いた．データの垂直分割は図 5 に示したように行う．ここでは簡単のために 2 台の PE を用いていると仮定する．各レコードの属性 A，B，C は PE1 に割り当て，D，E は PE2 に割り当てる．このように属性を分割することにより，各 PE が担当するアイテム集合長が短くなり，共通集合演算の計算時間が短縮できる．

データの水平分割は属性空間探索と相性がよい組合せであるのに対し，データの垂直分割はレコード空間探索と相性がよい組合せである．データを垂直分割した場合，各 PE は異なる属性を割り当てられるが，割り当てられるレコードはすべての PE で共通であり，かつ元データのすべてのレコードが含まれている．これは，各 PE でレコードの組合せを作る際に他の PE からレコード情報を受け取らずにすべてのレコード組合せを生成できることを意味する．

以下，本手法の特徴を図 6 のサンプルデータベースを元に処理例を通して説明する．

図 6 は，図 5 のように関係データベースを垂直に二分割し，それぞれをトランザクションデータベースに変換したデータである．分割前のデータは，3.1 節の逐次版の例題で用いた図 3 のデータである．図 6(a) は PE1 が (b) は PE2 が処理すると仮定する．例では，最小サポートカウントを 3，最小パターン

(注1)：木構造をパス 4 まで伸ばせばノード (0234, ABE) が生成されることがわかる．

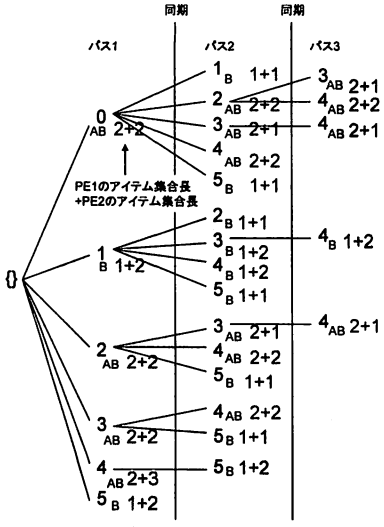

図 7　PE1 の処理例

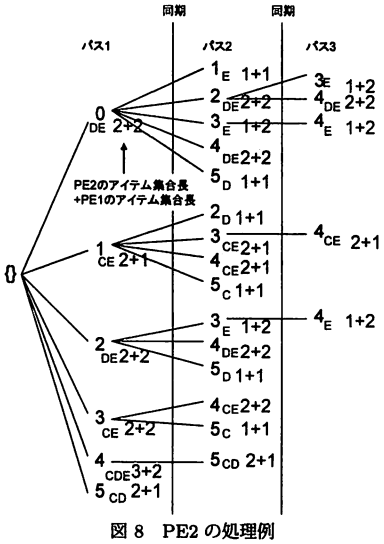

図 8　PE2 の処理例

長を 3 と設定した．

図 7 は PE1 の処理例，図 8 は PE2 の処理例である．処理の基本的な流れは逐次版と同様であるが，並列版では最小パターン長によるノード除去の前に PE 間でパターン長の同期処理が入る．例えば，パス 2 における同期処理を考えてみる．図 7 からパス 2 の各ノードの持つアイテム集合長を上から並べると 1,2,2,2,1,1,1,1,1,2,2,1,2,1,1 である．この段階で最小パターン長 3 に満たないといってすべてのノードを除去するのは誤りである．なぜなら PE1 と PE2 に属性を分割したため分割前のパターン長を求めるためには PE 間でパターン長を合計する必要があるためである．図 8 からパス 2 の各ノードのアイテム集合長を上から順に並べると 1,2,1,2,1,1,2,2,1,1,2,1,2,1,2 である．互いにアイテム集合長をブロードキャスト (注2) して和を取ると 2,4,3,4,2,2,3,3,2,3,4,2,4,2,3 となる．これは，逐次版の処理例である図 4 のパス 2 のアイテム集合長と等しくなる．後は，逐次版の手順どおりに最小パターン長 3 に満たないノードを除去すればよい．このアイテム集合長の同期はパスごとに行う必要があるが，各 PE が生成する木構造はまったく同じであるためアイテム集合長のみブロードキャストすればよい．

(注2)：MPI_Allreduce 関数を用いた．

表 2　テストデータセット

データ名称	平均レコード長	レコード数	アイテムの種類
T4000.I4.D50	4000	50	100000
T4000.I4.D100	4000	100	100000
Hyperdip50	12625	64	126250

最後に最小サポートカウントのパス 3 に含まれるアイテム集合をブロードキャストして完全なアイテム集合を生成する．例えば，PE1 で残ったアイテム集合は，上から AB, AB, AB, B, AB である．一方，PE2 で残ったアイテム集合は，上から E, DE, E, CE, E である．それぞれ和集合を取ると，ABE, ABDE, ABE, BCE, ABE となり，これは逐次版のパス 3 に含まれるアイテム集合と同じである．後は逐次版と同様の方法で頻出パターンを求める．

4. 実験結果と考察

本章では，提案手法をテストデータセットに適用し，実行時間を測定した結果について述べる．実験に用いたのは，CPU が Xeon 2.4GHz，メモリが 2GB，仮想メモリが 1GB の 16 台構成の PC クラスタである．プログラムは C++と MPICH2 ライブラリ (注3) を用いて作成した．

4.1 テストデータセット

表 2 は実験で使用したテストデータセットである．T4000.I4.D50，T4000.I4.D100 は，IBM Quest Synthetic Data Generation Code (注4) を用いて生成した．指定したパラメータ以外はデフォルト値である．Synthetic Data では，アイテム数を一定にし，平均レコード長を長くすることで高次元データを表現した．これは，多値属性は必ず離散化したどれか 1 つの値を取るため属性数に比例してレコード長が長くなると考えたためである．Hyperdip50 は，現実のマイクロアレイデータセット (注5) である．各属性値は連続値であるため等間隔区間離散化手法により各属性を 10 分割して離散化した．予備実験として，表 2 に対して CD を適用してみたが，メモリオーバーしてクラッシュしたため計算できなかった．

各データセットは，2 から 16 の PE に垂直分割して配置するが，データ分割方法にはブロック分割を用いた．例えば，T4000.I4.D50 はアイテムの種類が 100000 個ある．これを 4 台にブロック分割する場合，0-24999 の範囲を PE1，25000-49999 の範囲を PE2，50000-74999 の範囲を PE3，75000-99999 の範囲を PE4 に割り当てた．

提案手法では，最小サポートカウントのほかに最小パターン長を設定する必要がある．本実験の主目的は台数効果を測定す

(注3)：http://www-unix.mcs.anl.gov/mpi/mpich/
(注4)：http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/mining.shtml
(注5)：http://www.stjuderesearch.org/data/ALL1/

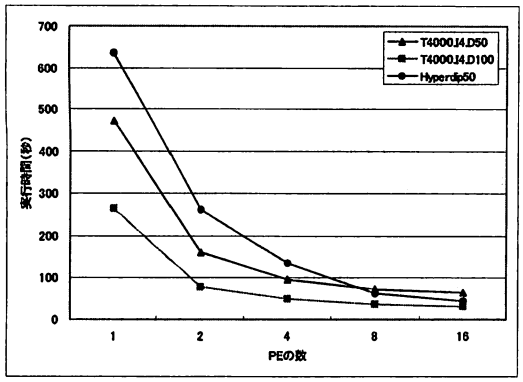

図 9 実行時間

ることにあるため各データの最小サポートカウントと最小パターン長は，データの特徴に合わせて適当に設定した．最小サポートカウントや最小パターン長の設定による計算時間の変化は [7] を参照されたい．

4.2 評価実験

まず台数効果を調べた．図 9 は各データセットの計算時間を測定したグラフである．横軸は PE の数，縦軸は実行時間である．実行時間は，

実行時間を調べるといくつかのケースでスーパーリニア効果が確認できた．スーパーリニア効果とは N プロセッサを用いて並列化した場合に N 倍以上の速度向上が得られる現象である．例えば，T4000.I4.D50 では，PE 数 2 のとき 2.86 倍，PE 数 4 のとき 4.64 倍，T4000.I4.D100 では，PE 数 2 のとき 3.38 倍，PE 数 4 のとき 5.27 倍，Hyperdip50 では，PE 数 2 のとき 2.39 倍，PE 数 8 のとき 8.82 倍の速度向上が得られている．スーパーリニア効果は，複数の PE を利用することでキャッシュ容量が増加し，キャッシュが効率的に使われたためだと解釈できる．

図 10 は，ノード生成部，ノード除去部，プロセス間通信部ごとに計算時間を累積したグラフである．複数の PE を用いている場合には，PE1 から計算時間を取得した．このグラフから，並列化による計算時間短縮の原因はアイテム集合長の共通集合演算を含んだノード生成部にあることがわかる．各 PE に属性を分割して配置したため各 PE のアイテム集合長が短くなり，共通集合演算が高速化したことがわかる．

図 11 は，全計算時間のうち各処理部が占める割合を図示したグラフである．このグラフから PE 数を増やすにしたがって，プロセス間通信の計算時間が増加していることがわかる．しかし，プロセス間通信の占める割合は，PE 数 16 でも 10%以下であり，並列化効率の高い手法であることがわかる．

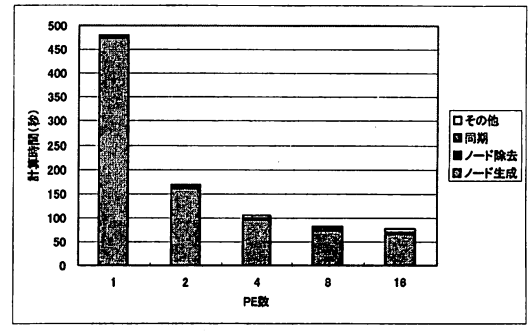

図 10 各計算部の処理時間の累積グラフ

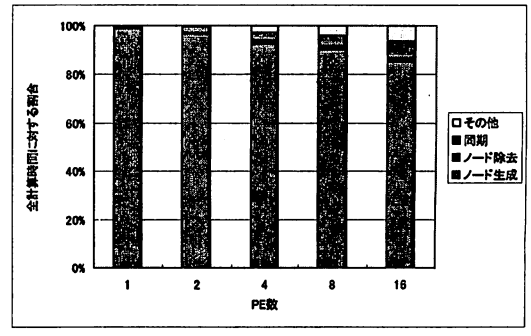

図 11 各計算部の処理時間の全計算時間に占める割合

5. おわりに

本論文では，レコード数が少なく，属性数が極めて大きいデータを対象にした並列頻出パターン発見手法を提案した．本手法は，データの垂直分割とレコード空間探索の組合せた並列アルゴリズムがプロセス間通信を抑え，超高次元データの頻出パターン発見に有効であることを示した．今後の課題としてデータベースの属性分布を考慮した属性の均等配置手法の検討などが考えられる．

文献

[1] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, In Proc. of the Intl. Conf. on Very Large Data Bases, pp.487-499, 1994.

[2] R. Agrawal and R. Srikant, Parallel Mining of Association Rules, IEEE Trans. on Knowledge and Data Engineering, Vol.8, No.6, 1996.

[3] T. Shintani and M. Kitsuregawa, Hash Based Parallel Algorithms for Mining Association Rules, In Proc. of 4th Intl. Conf. on Parallel and Distributed Information Systems, pp.19-30, 1996.

[4] J. S. Park, M. Chen and P. S. Yu, Efficient Parallel Data Mining for Association Rules, In Proc. of the ACM Intl. Conf. on Information and Knowledge Management, pp.31-36, 1995.

[5] A. Savasere, E. Omiecinski and S. Navathe, An Efficient Algorithm for Mining Association Rules in Large Databases, In Proc. of 21st Intl. Conf. on Very Large Databases, pp.423-444, 1995.

[6] D. W. Cheung, J. Han, V. T. Ng, A. W. Fu and Y. Fu, A Fast Distributed Algorithm for Mining Association Rules, In Proc. of 4th Intl. Conf. Parallel and Distributed Information System, 1996.

[7] 森 紘一郎, 折原 良平, 超高次元データからの頻出パターン発見手法, DBSJ Letters, Vol.6, No.1, 2007.