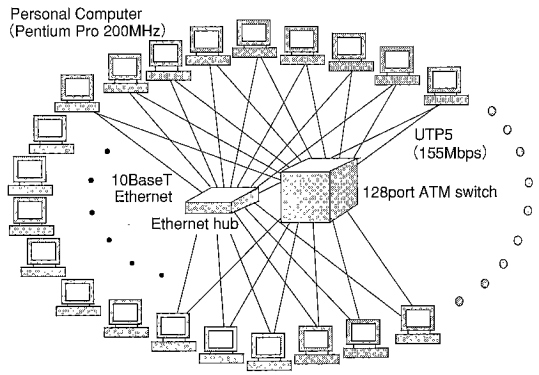
データベースとデータマイニングにおける並列処理

喜連川　優［東京大学］

はじめに

今日，企業においてはトランザクション処理を中心とする基幹系データベースとは別に戦略的意志決定を支援すべくデータウェアハウスの構築が進められている．各部門ごとの個別のデータベースやそれ以外の多様なデータを統合的に一元管理することにより，より大局的な問合せを実行したり，高度な経営分析をすることを目的としており，テラバイト級のデータウェアハウスが多く運用されるに到っている．データウェアハウス，すなわち巨大データベースに対する処理は容易に予想されるようにきわめて処理負荷が高く，常に高性能化への探究が続けられている．現時点では高価なメインフレームや専用サーバが用いられているが，安価なパソコンクラスタ（PCクラスタ）による大規模データベースサーバ実現の可能性を明らかにすべく実験システムを構築した．データウェアハウスの利用者ツールとして，莫大なデータからのルール抽出を行うデータマイニング，ならびに多次元キューブ上での視覚的操作によるデータ解析を支援するOLAP（On-Line Analytic Processing）などの手法が最近大きく注目されている．データマイニングは，その対象とするアプリケーションの特性により適用する手法も異なるが，いずれも大規模なデータを対象とする場合には多大の時間を必要とする．データマイニング操作は，今後データベースにおける基本演算として定着すると予想されることから，PCクラスタ上において相関ルールマイニングの並列化を試みた．本稿では，PCクラスタによる並列データベース処理，ならびに並列データマイニング処理について報告する[2), 3), 5)]．



CPU	Intel Pentium Pro 200MHz
Chip Set	Intel 440FX
Main Memory	64MB
Disk Drives	For OS Western Digital Caviar 32500 (EIDE, 2.5GB)
	For databases Seagate Barracuda (Ultra SCSI, 4.3GB, 7200RPM)
SCSI HBA	Adaptec AHA 2940 Ultra Wide
ATM NIC	Interphase 5515 PC ATM Adapter
OS	Solaris2.5.1 for x86

図-1　NEDO-100 PCクラスタの構成

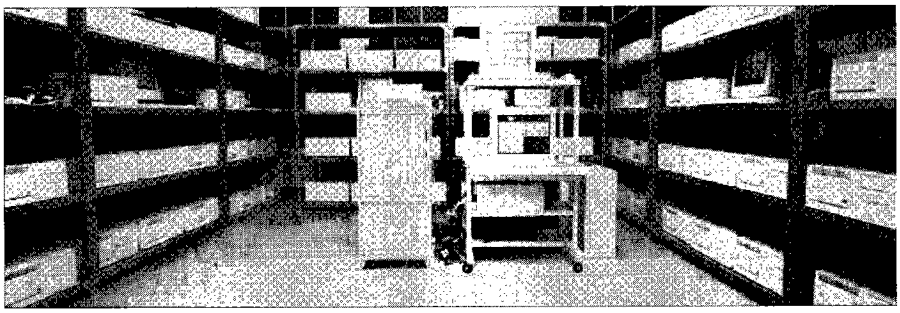


図-2　NEDO-100概観

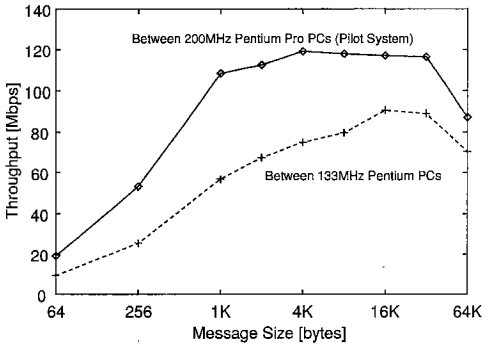


図-3　PCクラスタの通信性能

100ノードPCクラスタの構成

• 全体像

図-1に実験システムNEDO-100の構成図，図-2にその概観を示す．Pentium Pro（200MHz）プロセッサ，64Mバイトの主記憶，IDEならびにSCSIのディスクからなるパソコンを1ノードとし，100ノードをATMスイッチにより結合している．昨今のPCクラスタプロトタイプのほとんどがMyrinet，あるいは100Mイーサを利用しているのに対し，ATMを選択した．本プロジェクトは1995年秋に認められたものであるが，当時はATMへの期待感が強く通信のみならずコンピュータ相互接続への可能性も模索されていた．通常の科学技術計算アプリケーションではバンド幅に加えてレイテンシーを重要視するが，データベース処理は大容量データの定常的な流れにより形成されることから，レイテンシーよりもむしろ高いバンド幅が望まれ，ATMに適したアプリケーションとして当初よりATMの採用を念頭に置いてきた．図-3にノードペア間の通信性能を示す．Pentium Proでは，適切なブロックサイズを選択することにより，TCP/IPプロトコルで120Mbit/秒程度の実効転送速度を達成可能なことが分かる．

• コモディティアーキテクチャ

我々は本プロジェクトの前に，SDC（Super Database Computer）と名付けたデータベースサーバを構築してきた[1]．このマシンはMC68040 マイクロプロセッサを数十個利用したマイクロプロセッサベースの専用マシンである．負荷分散ならびに通信のための専用ハードウェアを実装している．大学を中心とした開発の常ではあるが，その開発期間はきわめて長期に及び，安定動作する頃にはマイクロプロセッサは数世代古い物となってしまっていた．もちろんプロセッサの性能，メモリ容量などで正規化することにより，SDCは他のマシンと比べてはるかに高い実行効率を発揮することは実証できたものの，開発期間の短縮は研究の観点からも製品開発の観点からも必須であるとの判断から，本プロジェクトでは，完全なコモディティを採用することとし，システム稼働時点でその絶対性能がハイエンドの商用システムを凌駕することを目標とした．すなわち計算主体のコモディティとしてのパソコン，そして当時近未来の通信のコモディティと予想されたATMスイッチを利用することとし，自前のハードウェアは一切開発しないこととした．SDCでは，基本ソフトウェアとしてリアルタイムカーネルVxWorks 5.1を利用していたが，NEDO-100ではSolaris 2.5.1 for x86を用い，通信も通常のTCP/IPプロトコルを用いている．VIAに代表されるようなゼロコピー通信プロトコルは，種々のデモがなされており，その動向が注目されるものの，後に述べるようにデータベース応用では，あるに越したことはないが，通常の通信機構で大きな問題はない．データベース処理ソフトはSDCプロジェクトにおいて開発したものを修正しつつ移植した．NEDOの支援による本プロジェクトは1995年の秋に認められ，調達の制約から機器の導入は1996年末となったが，1997年2月より100台のパソコンからなるマシンは本格稼働し始め，実際，開発期間は大きく短縮された．

• 関係データベース問合せの並列処理

1970年に関係データモデルが提案された後，関係データベース管理システム（RDBMS）は，80年代より商用化が始まり，今日きわめて広く利用されるに到っているのは周知の事実である．データベース処理は，大きく2つに分類することができる．小容量のデータの更新を高頻度に行うトランザクション処理は，銀行，座席予約，受注，在庫管理などに代表され，ビジネスの基幹業務において，きわめて重要な役割を果たしている．1つのトランザクションの処理負荷は，それほど大きくなく，多数のトランザクションを同時実行する際の並行アクセス制御が主要な技術的課題とされてきた．現在TPC-Cと呼ばれるベンチマークによってその性能が表されることが多い．これに対しもう1つのデータベース処理は，問合せ処理（Query Processing）と呼ばれ，データウェアハウスに対する検索を中心とする処理を指し，一般に多くのリレーションの走査，ジョイン（結合），複雑な集計を含むことが多い．このため，処理負荷はトランザクション処理に比べるとはるかに重くなることが通例である．また処理時間が長いため，それほど頻繁に問合せが発行されることはない．本プロジェクトでは，後者の高性能化，すなわち1つの重いデータベース処理を並列プロセッサによって高速化することを目的とした．

関係データベースの並列化の最大の魅力は，データ操作言語SQLによって記述されている限りにおいて，ユーザはプログラムを変更する必要がなく，並列化を一切意識する必要がないという点にある．すなわち，科学技術計算のように並列化に苦しむ必要はない．一方その作業は，システムが行う必要があり，データの分割方式，負荷分散方式など多くの課題が残されている．

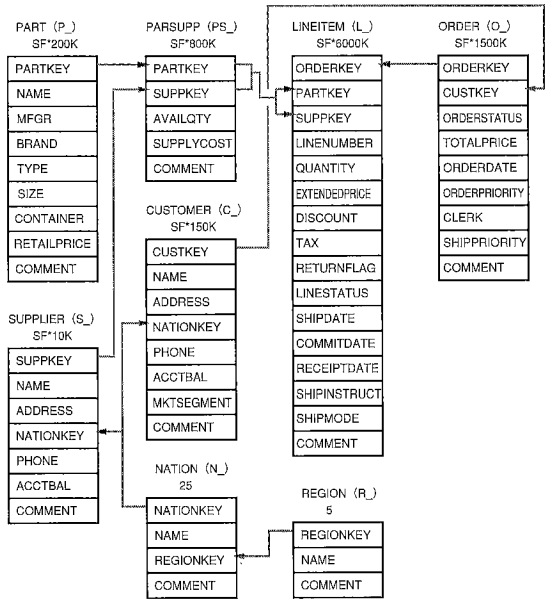


図-4　TPC-Dベンチマークにおけるデータベーススキーマ

並列データベース問合せ処理については，従来，種々の研究がなされてきた．とりわけ，関係データベースでは，2つの関係表を動的にジョインすることにより柔軟な検索が可能となる点に大きな特徴があるものの，ジョイン演算処理負荷はきわめて高く，現在でもその高速化に関する研究は続けられている．NEDO-100では従来より我々が研究を進めてきたハッシュ結合技術を実装している．

• TPC-Dベンチマーク

1995年にTPC（Transaction Processing Council）はTPC-Dなる意志決定問合せ処理のためのベンチマークを策定した．TPC-A，TPC-B，TPC-C（現在はA，Bは利用されていない）は，すべてトランザクション用データベースベンチマークであり，TPC-Dは最初の問合せ処理用ベンチマークである．TPC-Dは現在，改版が計画されているが，8つのリレーションに対し，17個の問合せが規定されている．この中で9番目の問合せ（Q9）は，5つの関係表のジョインが必要となり，最も処理負荷が大きく，キラークエリと呼ばれている．データベーススキーマを図-4に，またQ9のSQL表現を図-5に示す．

絶対時間の比較は，システム資源のコストで正規化する必要がある．TPCではコストを明確に規定している．一方，大学で開発した実験システムはコストが必ずしも正確ではないため，商用システムとの比較は容易ではないが100GBベンチマークに関し，Q9はたとえば，1997年9月27日までは最高パワー値を有するシステムにおいて953.3秒，それ以降執筆時点で927.8秒という値が最高値であるのに対し，NEDO-100上では177.2秒という値を達成しており，ハイエンドの商用システムに比する性能を達成するポテンシャルを有するといえよう．実行時間は，実行プランに依存する．ここではむしろ，相対性能をより詳細に見ていくこととする．図-6に1つの実行プランを示す．図-7に当該プランの実行時の各種資源の利用率（CPU利用率，ディスクからのデータ転送レート，ネットワークを介したデータの送受レート）の様子を示す．図-7に示されるごとく，処理負荷は演算により異なるため，部分的にCPUバウンドであったりI/Oバウンドであったりしている．より高速化すべくファイル構造をトランスポーズ化することを試みた．図-8にトランスポーズドファイルを用いた実行木を示す．トランスポーズドファイルでは，アトリビュートごとにファイル化することにより必要とするアトリビュート群のみを取り出すことが可能となるため，通常のレコード形式のファイル編成に比べ，大幅にディスク入出力量を低減させることが可能となる．実際，Q9ではディスク入出力を20％に低減することが可能となる．一方，アトリビュート間を結合するためにTID（Tuple ID）ジョインを多数回実行する必要がありCPUコストが増大する．図-8からもその処理が大きく複雑化していることが予想されよう．また，図-9に示される実行の様子から明らかなようにシステムは全実行過程において，完全にCPUバウンドとなっている．従来，トランスポーズドファイルは，CPUパワーを多く必要とすることから現実性がなかったが，近年のマイクロプロセッサの急速な性能の向上に伴い，そこから得られる過剰なCPUパワーをより有効に活用することにより，CPU処理コストが増加してでも入出力量を低減した方が高い性能が得られることが分かる．実行時間は，177秒から94秒へと低減し，大幅な性能向上を達成可能であることが分かった．今後，より高速なCPUが利用可能になれば，あるいはSMPタイプのパソコンを利用すれば，一層の高速化が可能である．

```
select Nation, Year, sum (Amount) as Sum_Profit
from (select N_Name as Nation,
        extract (year from O_Orderdate) as Year
        L_Extendedprice * (1 - L_Discount)
          - PS_Supplycost * L_Quantity
            as Amount
      from part, supplier, lineitem,
           partsupp, order, nation

      where S_Suppkey  = L_Suppkey
        and PS_Suppkey = L_Suppkey
        and PS_Partkey = L_Partkey
        and P_Partkey  = L_Partkey
        and O_Orderkey = L_Orderkey
        and S_Nationkey = N_Nationkey
        and P_Name like '%green%'
    )
group by Nation, Year
order by Nation, Year desc
```

図-5　TPC-D Query 9

• 動的負荷分散

一部のノードが過負荷になると，十分に並列度を達成できないこととなる．ノード数の多い大規模システムでの最大の課題は，負荷分散にあるといえよう．巨

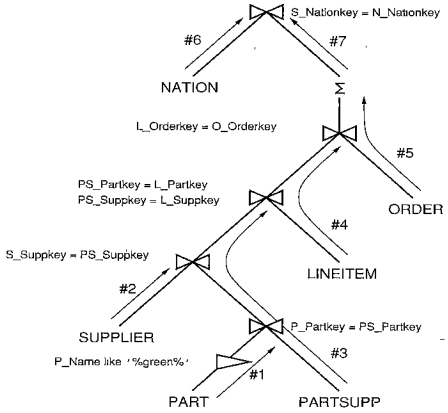


図-6　Query 9実行木

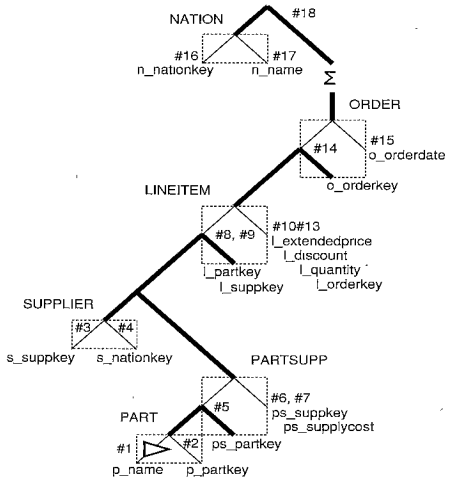


図-8　トランスポーズドファイルを用いた場合の実行木

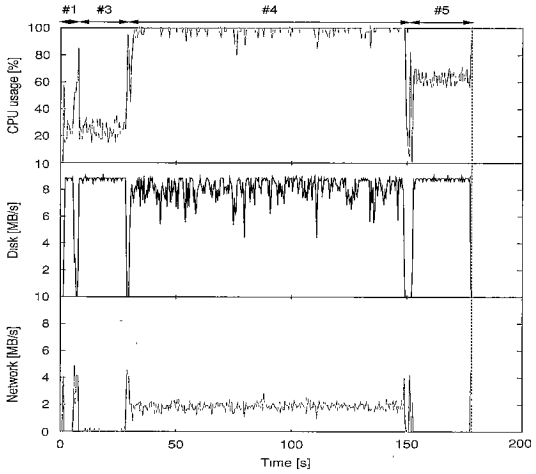


図-7　図-6に示される実行木に基づき実行した時の様子

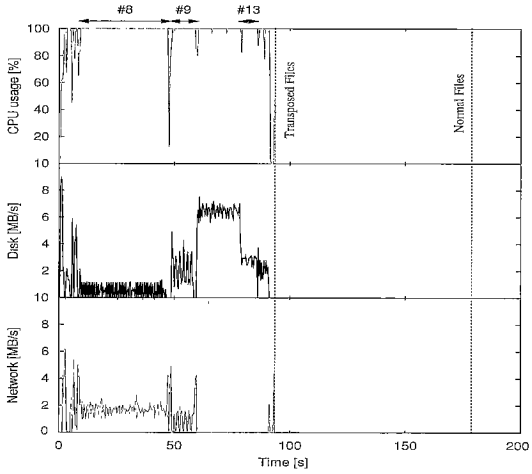


図-9　トランスポーズドファイルを用いた場合の実行の様子

大データベースを対象とする場合，実行前の静的な負荷推定は困難であるため，システム規模が大きくなるにつれて，実行時動的負荷分散が重要な技術となると考られる．ここでは，ジョイン演算の動的負荷分散を試みた[4]．データに偏りが存在すると，特定ノードに処理が集中することから，当該ノードが過負荷となる．詳細は略すが，ハッシュエントリのプローブ頻度を負荷と捉え，高負荷ノードより低負荷ノードにハッシュテーブルエントリを移送し，負荷が全ノードで均衡化するように調整することにより，負荷分散を試みた．現在の実装では1つのマスタノードが各プロセッサからその負荷状況を定期的に収集し，偏りがしきい値を超えると負荷分散を試みる．**図-10**に動的負荷分散の有無を比べることによりその有効性を示す．図-10（a）では，1つのノードに負荷集中が生じていることを示し，その他のノードではほとんどCPUが利用されていない．一方図-10（b）では，ハッシュエントリの移送により，すべてのプロセッサの利用率が向上し，同時に実行時間が短縮していることが分かる．

相関ルールマイニングの並列処理

トランザクション処理によって生成される莫大なログデータは，従来回復処理以外の目的では利用されることはなかった．近年，プロセッサ能力，ならびに二次記憶容量の増大を背景に大容量データの解析が可能となり，膨大なデータの中から有意なルールの抽出を試みるデータマイニングが注目されている．小売業ではバスケットアナリシスと呼ばれる顧客の購買動向の解析手法が広く利用されつつある．これは「パンとバターを買う客は，その90％がミルクも購入する」に示されるような形のルール（相関ルール）を抽出する手法であり，スーパーマーケットにおける商品の並べ方や値引き品目の決定などに活用することができる．この他にも時系列パタンからのルール抽出など，種々のマイニング手法が研究されている．相関ルールに関しては，IBMアルマデン研究所よりアプリオリと名付けられた高速アルゴリズムが提案され[6]，以降その改良が続けられている．ここでは，アプリオリを並列

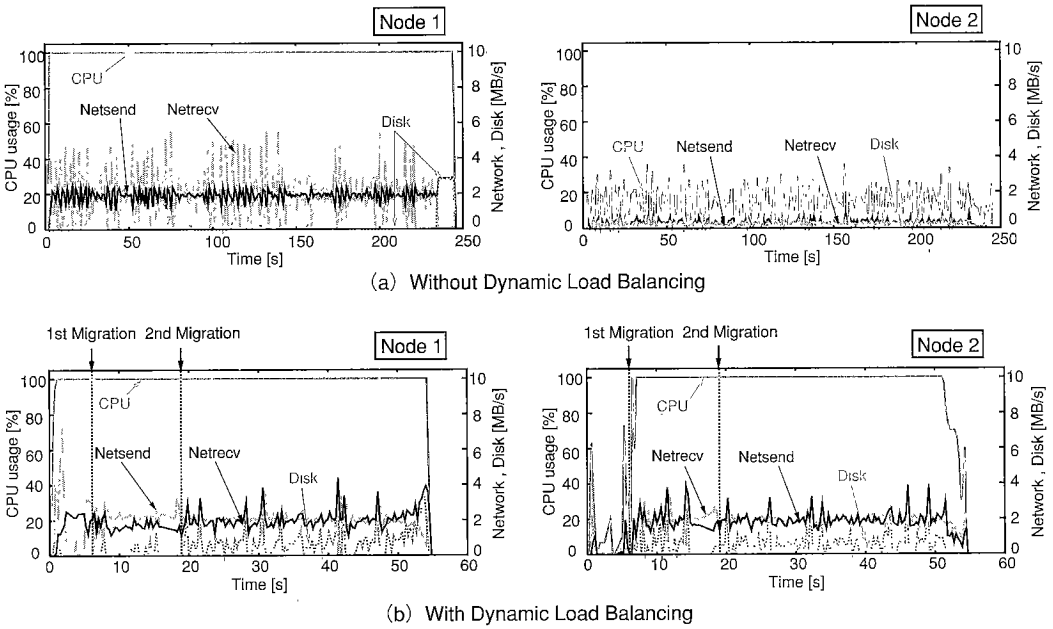


図-10　実行時動的負荷分散機構の効果

化したアルゴリズムをPCクラスタ上に実装した．

• 相関ルール

アイテムの集合を$I=\{i_1, i_2, \ldots i_m\}$，トランザクションデータベースを$D=\{t_1, t_2, .. t_n\}$ $(t_i \subseteq I)$ とする．ここで，各要素t_iをアイテム集合（itemset）と呼ぶ．長さkのアイテム集合とはk個のアイテムの組合せを指す．アイテム集合Xの支持度*support*（X）は，D全体に対しXを含むトランザクションの割合を表す．

相関ルール（association rule）は，$X \Rightarrow Y$で表現され（$X, Y \subseteq I$，$X \cap Y = \phi$），支持度（support），確信度（confidence）なる2つのパラメータによりその有意性を示す．相関ルール$X \Rightarrow Y$の支持度*support*（$X \Rightarrow Y$）は，D全体に対しXとYを共に含むトランザクションの割合*support*（$X \cup Y$）により，また，確信度*confidence*（$X \Rightarrow Y$）はDの中でXを含むトランザクションの中で，XとYを共に含むトランザクションの割合，すなわち*support*（$X \cup Y$）/*support*（X）によって定義される．

相関ルールの抽出問題はユーザによって指定された最小支持度（minimum support）と最小確信度（minimum confidence）を満足するすべてのルールを見出すことに相当する．

相関ルールは次の2つのステップで抽出される．

(1) 最小支持度を満足するアイテム集合をすべて見出す．これらのアイテム集合をラージアイテム集合と呼ぶ．

(2) 第1ステップで求めたラージアイテム集合から最小確信度を満たす相関ルールを導き出す．

相関ルール抽出処理のうち，第2ステップは軽いのに対し，第1ステップはトランザクションデータベースを繰り返し走査し，負荷は重く処理時間の大半を占めることになる．第1ステップの効率化を試みる研究が中心となっている．

ラージアイテムセットを効率良く求めるアルゴリズム，アプリオリの概略を示す．

(1) 長さ（$k-1$）のラージアイテム集合から長さkの候補アイテム集合を作成する．

(2) トランザクションデータベースを検索し，支持度を求める．

(3) 最小支持度を満足するものを取り出し，長さkのラージアイテム集合を求める．

上の長さkのラージアイテム集合を求める処理をパスkと呼ぶ．この処理は新たなラージアイテム集合が空となるまで続けられる．ここで，長さ（$k-1$）のラージアイテム集合から長さkの候補アイテム集合を作成する際に，種々の手法により絞り込みを行う．

• 並列相関ルール抽出技法

アプリオリの最も簡単な並列化は，トランザクションデータベースを分割し各ノードで独立に候補アイテム集合の出現頻度を求める方法（Count Distribution）[7] であるが，この手法では，候補アイテム集合をすべてのノードに複製する必要がありメモリ効率が著しく低い．また，パス2の候補アイテム集合はサポート値の設定によってはラージアイテム集合の1000倍程度も肥大化することがあり[8] 処理時間の大半は2番目のパスに費される場合が多い．メモリオーバフローが生ずる可能性があるため，それを低減すべくハッシュを用いて候補集合を各ノード群に分散するHPAなる手法を用い並列化を試みた[9]．それぞれのパスで各ノードは自ノード内のトランザクションデータベースを読み，アイテムセットを生成し，当該アイテムセットを管理しているノードへ送信する．各ノード

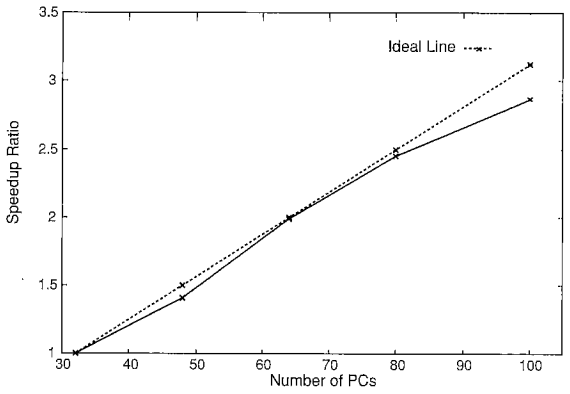


図-11　並列相関ルールマイニングの処理性能

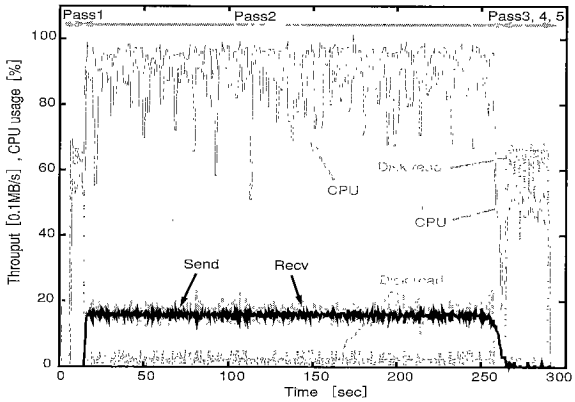


図-12　並列相関ルールマイニングの実行の様子

は，自己宛のアイテムセットを受信すると，数え上げを行う．**図-11**に1000万件のトランザクションデータベースを用いた並列相関ルールマイニングにおける台数効果を示す．高い並列度が得られていることが分かる[10]．**図-12**に，横軸を経過時間とした際の各種資源の利用率を示す．パス2はCPUバウンドとなり，長時間要していることが分かる．データベース処理に比べると，データマイニングではアイテムセットの切り出しや候補の枝刈り処理などにより，多くのCPUサイクルが必要となっている．ここでは誌面の都合上，説明できないが，より質の高いルールを抽出するために分類階層を導入した相関ルールマイニングも行われており，そこでは一層処理負荷が増大する傾向がある[11]．

おわりに

筆者が学生の頃は，マイクロプロセッサは量産すればただ同然になるので，それを好きなだけ使って強力なマシンを作れないかという並列マシンアーキテクチャの研究が盛んであった．今やプロセッサのチップだけではなく電源，ファン，匡体，OSまで含めたパソコンを1つのパーツとみなす時代になったといえる．本稿では，データベースならびにデータマイニングというアプリケーションを取り上げ，パソコンをパーツとして構築した大規模PCクラスタ上での実装について報告した．科学技術計算を中心とする数値計算応用以外のアプリケーションにもPCクラスタは有効である．プロジェクト当初，Q9は$17Mの最上位システムで953秒，$2.1Mのシステムで2639秒必要としており，NEDO-100は1997年春時点において，177秒まで高速化を達成した[2]．当時筆者の知る限り，100ノードのPCクラスタ上でのデータベース実験報告は聞いていない．ベンチマーク値は刻々と更新される．同年9月以降もQ9の性能は大きく変化していないが，価格は大幅に下がり他の問合せは全体的に高速化されたが，それでも本システムは17個の内，9個の問合せで優位性を保っている．また，データマイニングの並列化の本格的研究は始まったばかりである．

データウェアハウスは通常，従来の基幹系（レガシー）システムとは独立して構築される場合が多く，今後低コストなPCクラスタへの移行が進むものと予想される．また，ERP（Enterprise Resource Planning）をはじめとするより強力なパワーを必要とするアプリケーションへの挑戦も興味深い．情報洪水の中で今後ますます情報の管理，検索にCPUパワーを注入することが不可欠と考えられる．PCクラスタはデータベース技術にとって必須のプラットフォームとなろう．

参考文献

1) Tamura, T., Nakamura, M., Kitsuregawa, M. and Ogawa, Y.: Implementation and Performance Evaluation of The Parallel Relational Database Server SDC-II, Proceedings of International Conference on Parallel Processing '96, pp.212-221 (Aug. 1996).
2) Kitsuregawa, M., Tamura, T. and Oguchi, M.: Parallel Database Processing/Data Mining on Large Scale ATM Connected PC Cluster, Parallel and Distributed Systems Euro-PDS '97, pp.313-320 (June 1997).
3) Tamura, T, Oguchi, M. and Kitsuregawa, M.: Parallel Database Processing on a 100 Node PC Cluster: Cases for Decision Support Query Processing and Data Mining, ACM SC97: High Performance Networking and Computing (1997).
4) 安井隆宏, 田村孝之, 小口正人, 喜連川優: 並列DBMSにおける動的負荷分散機能: 負荷情報収集に関する一考察, 情報処理学会, データベース研究会研究報告, Vol.98, No.57, pp.87-94 (July 1998).
5) Tamura, T., Oguchi, M. and Kitsuregawa, M.: High Performance Parallel Query Processing on a 100 Node ATM Connected PC Cluster, IEICE Trans on Information & Systems, to appear (Jan. 1999).
6) Agrawal,R. and Srikant, R.: Fast Algorithms for Mining Association Rules, Proceedings of VLDB, pp.487-499 (1994).
7) Agrawal,R. and Shafer,J.C.: Parallel Mining of Association Rules: Design, Implementation and Experience, IBM Research Report RJ 10004 (1996).
8) 喜連川優: データマイニングにおける相関ルール抽出技法, 人工知能学会誌, Vol.12, No.4, pp.513-520 (July 1997).
9) Shintani, T. and Kitsuregawa, M.: Hash Based Parallel Algorithms for Mining Association Rules, Proceedings of IEEE Fourth International Conferrence on Parallel and Distributed Information Systems, pp.19-30 (Dec. 1996).
10) Oguchi, M., Shintani, T., Tamura, T. and Kitsuregawa, M.: Optimizing Protocol Parameters to Large Scale PC Cluster and Evaluation of its Effectiveness with Parallel Data Mining, HPDC-7 (July 1998).
11) Shintani, T., Kitsuregawa, M.: Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy, Proceedings of ACM SIGMOD International Conference on Management of Data, pp.25-36 (June 1998).

（平成10年9月10日受付）