卒業論文

PC クラスタ上でのデータマイニングの並列化の検討

氏　名	：	徳永　陽平
学籍番号	：	2210990405-1
指導教官	：	山崎　勝弘　教授
提出日	：	2002 年 2 月 21 日

立命館大学　理工学部　情報学科

内容梗概

本論文では,PC クラスタによるデータマイニングの並列化の検討について述べる.

情報化社会の現代では情報がとても重要なものとなっている.企業などでは日々さまざまな方法を使いデータを集め,そのデータの中から有益である情報を抽出し今後の戦略に生かそうとしている. この有益である情報を抽出するのがデータマイニングである.データマイニングで扱うデータはほとんどが大規模なものである.そのため処理時間がかなりかかる場合もあり,それが問題になる.そこで本研究では PC クラスタを使用し,並列処理を行うことで処理時間の問題の解決を図る.

本論文では,データマイニングのデータの変換,簡易化部分を作成し,出力した結果の統計をとる.さらにその部分の並列化の検討を行う.

目次

図目次

表目次

１．はじめに

情報化社会の現代では情報がとても重要なものとなっている.企業などでは日々さまざまな方法を使いデータを集め,そのデータの中から有益である情報を抽出し今後の戦略に生かそうとしている. このデータから有益である情報を抽出する事をデータマイニングで行う. データマイニングで扱うデータは膨大なものである.なぜなら,情報技術の進歩により情報収集技術が向上し,さらに記憶装置の低価格化により、データの収集,保存の能力が上がったためである.

データマイニングで膨大なデータを扱う場合にはいくつかの問題がある.その中の一つがデータの量的な問題である.データの量的な問題は,蓄積されたデータが膨大になりすぎて起きてしまう問題である.データの収集は日々行っているため新しいデータが増えていく一方である.だからといってデータが膨大になるのを回避するために,データマイニングで使用するデータから古いデータを削除し新しいデータを増せばいいわけではない.データは新しい方が価値が高く古いデータの価値は時間とともに減少していく.しかし,古いデータだけを見れば価値のないものとなっていても,現在のデータと過去のデータをあわせることにより,現在の位置や今後の方向性などの情報が読み取れる場合もある.つまり,データを減らすことによってデータの重要な部分が失われ,重要な分析の機会も失ってしまう場合もある.そのため,データの量的な問題を解決するためにデータを削除するというのはあまり良い方法ではない.データの量を気にせずに処理するには処理効率を向上させる必要がある.そこで処理効率の向上を図るために並列処理を使い問題解決を行う.

並列処理は現在さまざまな分野で浸透してきている.今では様々な高性能計算機が製品化され,価格性能比も飛躍的に向上している.実際に並列計算機は科学技術計算や画像処理,実時間シミュレータなど,大規模な計算で活用され,大きな成果をあげている.近年複数の PC を接続した PC クラスタが注目されている.PC2 台から実現できる PC クラスタは,予算に応じて,個人で持つことのできる並列コンピュータである.この PC クラスタの出現,さらに,PC の価格性能比の向上により特権的にしか使えなかった,スーパコンピュータ環境を誰でも使えるようになった.そのような点から PC クラスタを使用した並列処理を使い大規模なデータ処理の処理効率の向上を行う.

本研究では PC クラスタを使いデータマイニングの並列化の検討を行っている.

第 2 章では並列処理で理解しておかなければならない事柄を,第 3 章ではデータマイニングについての概要,手法などについて説明する.そして第 4 章ではデータマイニングの並列化の検討と今回実際作成したデータマイニングの前処理部分の実行結果を記述する.

２．並列処理

2.1 並列処理の概要

並列処理は高速性を追求するために考え出されたものである.それは名前のごとく１つの仕事を分割し複数のタスクで並列に実行する事で高速に処理を行う.そのため 1 台で仕事を行うよりも 2 台のほうが 2 倍,4 台のほうが４倍速くなると考えられる.並列に実行している部分はそうなるであろう.しかし実際にはそれは理想的な並列処理であり,実際にはそうならない場合が多い.このような並列処理による効果を妨げる要因として,逐次処理部分のオーバヘッド,負荷のばらつき,通信/同期のオーバヘッドの３つが挙げられる.[1]

１．逐次処理部分のオーバヘッド

有名なアムダールの法則によれば､逐次処理部分の割合が全体の s(0≦s≦1)存在すれば n 台のプロセッサを用いて得られる速度向上は、

$$\frac{1}{s+\frac{1-s}{n}}$$

になる。例えば 1 台実行時に逐次部分の割合が 5%あれば、たとえ 10000 台のプロセッサを用いても得られる速度向上は高々20 倍なのである。この問題に対する対策としてはアルゴリズムを工夫することにより,並列部分の割合を大きくすることがあげられる.

２．負荷分散のばらつき

並列システムではｎ台のプロセッサを用いた場合、すべて並列処理可能であれば理想的には１／ｎに並列実行部分の処理時間を短縮できるはずである。しかしながら、実際には各プロセッサにおいて負荷のばらつきがある場合、並列実行部分の処理時間は最も遅いプロセッサに引っ張られることになる。そのため負荷の均衡のとれる分散方法が必要である.

３．通信／同期のオーバヘッド

並列プログラムでは、正しい解を得るためには逐次プログラムにはない,他のプロセッサとデータを通信すること,または全プロセッサで同期をとることが必要となる。このための時間は当然のことながら逐次プログラムにはないオーバヘッドとなる。

本論文で取り上げている OpenMP は共有メモリ用の並列プログラミング言語であり、各スレッドはスレッド間でアドレス空間を共有している。このためパイプやメッセージパッシングなどのプロセス間データ通信を使う必要がないため、通信のオーバヘッドは存在しない。しかしながら、同期のオーバヘッドは存在するのでこの問題も並列プログラミングをおこなう際には考慮しなければならい問題となる。

並列処理ではメモリの使い方が特徴的である.並列処理でのメモリの種類とその説明を以下に示す.

（1）共有メモリ

複数のプロセッサがメモリバス/スイッチ経由,主記憶に接続される形態であり,メモリモデルとしては単純である(図 1).このアーキテクチャを有するシステムの事をSMP(Symmetrical Multi Processor)と呼ぶ.この形態は次のような利点がある.

- メモリモデルが最も汎用で,プログラムが組みやすい
- 並列処理だけでなく，スループットを重視するサーバマシン(逐次プログラム/プロセスを多数処理するマシン)としても適している

以上の理由により,近年ますます市場が増えてきている.

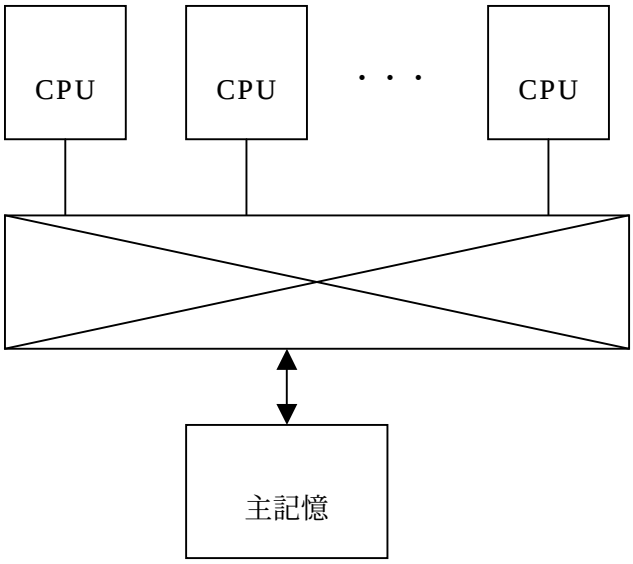


図 1：共有メモリモデル

（2）分散共有メモリ

プロセッサと主記憶から構成されたシステムが複数個互いに接続された形態である.ただし,プロセッサは他のプロセッサの主記憶の読み書きを行うことができる(図2).

この形態の特徴は,大規模なシステム構築が可能であるという事である.この形態と次に述べる分散メモリとのアーキテクチャの差は減ってきている.プログラミングモデルにおいては分散共有メモリのほうが自由度は高いが,1 個でもバリア同期の場所を間違えると,タイミング依存で非常に解析しにくいバグを作りこむ事になる.

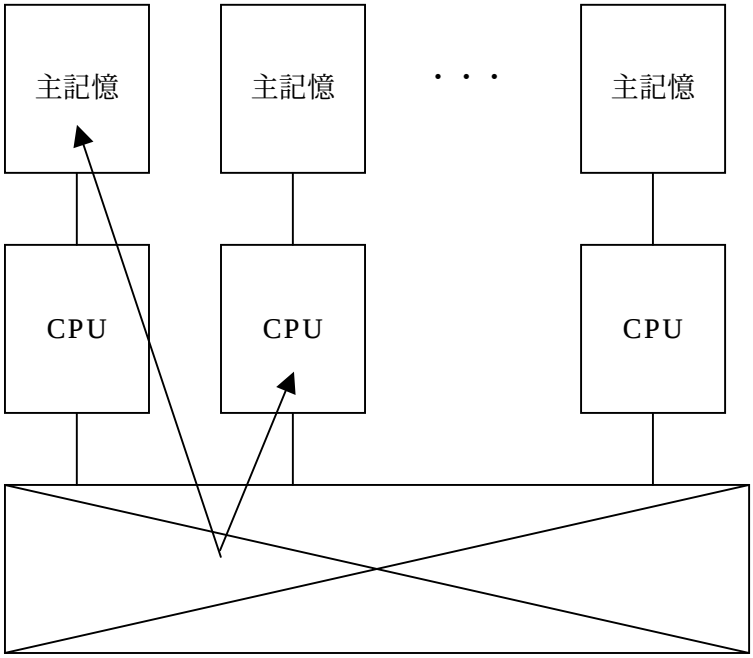


図 2：分散共有メモリモデル

（3） 分散メモリ

分散メモリはプロセッサと主記憶から構成されるシステムが複数個互いに接続された形態である.ただし,プロセッサは他のプロセッサの主記憶を読み書きを行う事ができない.必ず相手のプロセッサに介入してもらう必要がある(図３).

この形態の特徴として,大規模なシステムの構築が可能であるという事がある.また,通信遅延のオーバヘッドはきわめて高いが PC や EWS を LAN で接続したものもこの形態に属する.最近では安価なスイッチングハブが入手できるようになり,このような形態での並列プログラミングも盛んになりつつある.

この形態のプログラミングモデルは,デッドロックさえ気をつければ,タイミングに依存するいやなバグが発生する事は少ない.一方,通信をプログラミング時にすべ

てスケジュールするというのはプログラマにとって多大な負担となる.

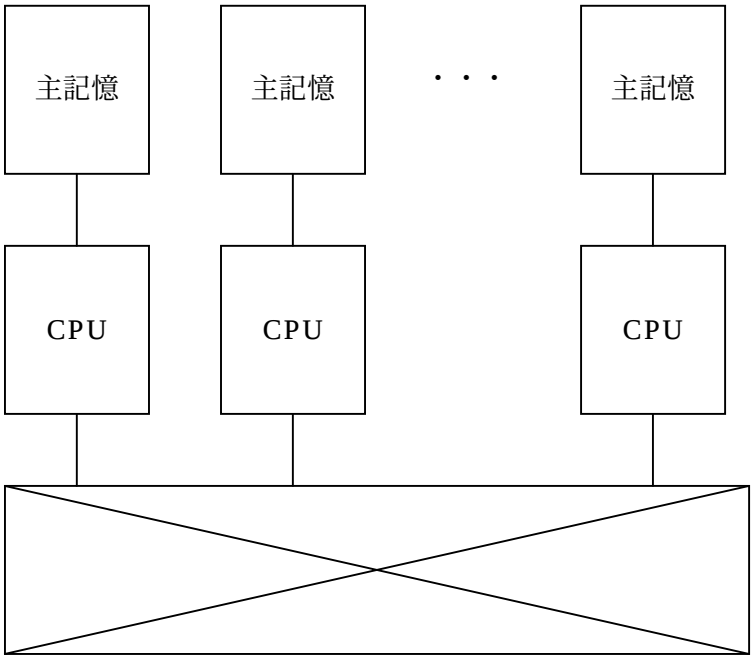


図 3：分散メモリモデル

2.2 並列アルゴリズム

並列アルゴリズムはプロセッサファーム(Processor Farms),分割統治法(Divide and Conquer), プロセスネットワーク(Process Networks), 繰り返し変換(Iteratibe Transformation)の大きく４つに分類する事ができる.以下にそれぞれの説明を書く.

(1) プロセッサファーム

まずマスターが問題を複数に分割する.それぞれをマスターもしくはスレーブが独立に処理を行い,結果をマスターに返す.マスターがそれらをまとめ最終的な解を得る.

(2) 分割統治法

問題に対し何らかの計算を行い,ある条件のもとにそれを分割する.そして分割された部分に対し,同様の計算を行いまた分割を行う.これを繰り返し条件が成立するまで再帰的に計算を行い,解を得る.問題の解は分割された部分解をまとめることにより得られる.

(3) プロセスネットワーク

まず問題を複数の計算ステージに分ける.各ステージは並列に実行されデータを計算ステージに順に流すことにより解を得る. この処理はパイプライン処理とも呼ばれる.

(4) 繰り返し変換

与えられた問題を複数のオブジェクトに分割する.各オブジェクトは複数の繰り返しの計算により値が変換される.その繰り返しはオブジェクトの値が条件を満たすまで繰り返し実行することで解が得られる.

2.3 PC クラスタ

コンピュータをネットワークでつなげて処理を行うシステムをクラスタと呼ぶ.特に PC を使って並列処理を行うシステムは PC クラスタと呼ばれている.PC を 2 台以上用意するだけで実現できる PC クラスタは,予算におおじて個人で持つことのできる並列コンピュータである.近年では PC の高性能化,低価格化が進み,今まで一部でしか使うことのできなかったスーパーコンピュータと同等の環境を PC クラスタで再現できるようになった[6].

本研究では RedHat Linux7.3 を OS に使用した表１のようなスペックの PC クラスタを使用している.

表 1 PC クラスタの性能

	サーバ	クラスタ 16 台
CPU:Pentium3	450MHz	500MHz
メモリ:SDRAM	512MB	512MB

これらを接続する Ethernet は最大帯域幅が 100Mbps,最小遅延は 80μsec である.クラスタ同士を接続する Myrinet は最大帯域幅が 2Gbps,最小遅延は 9μsec であり,メモリ空間を共有している.SCore の ver5.2.0 を使用することで SCore 型クラスタを実現している.さらに分散共有メモリを実現するために SCASH 等のソフトウェアを使用している(図４).

SCore 型クラスタとは RWC が開発した高性能通信ライブラリを持つクラスタ用のミドルウェアを使用し構築したクラスタのである.従来の Beowulf 型クラスタは低い通信性能のネットワークを使い,TCP/IP のプロトコルを使っているために通信性能に問題があった.その問題を SCore 型クラスタでは解決している.

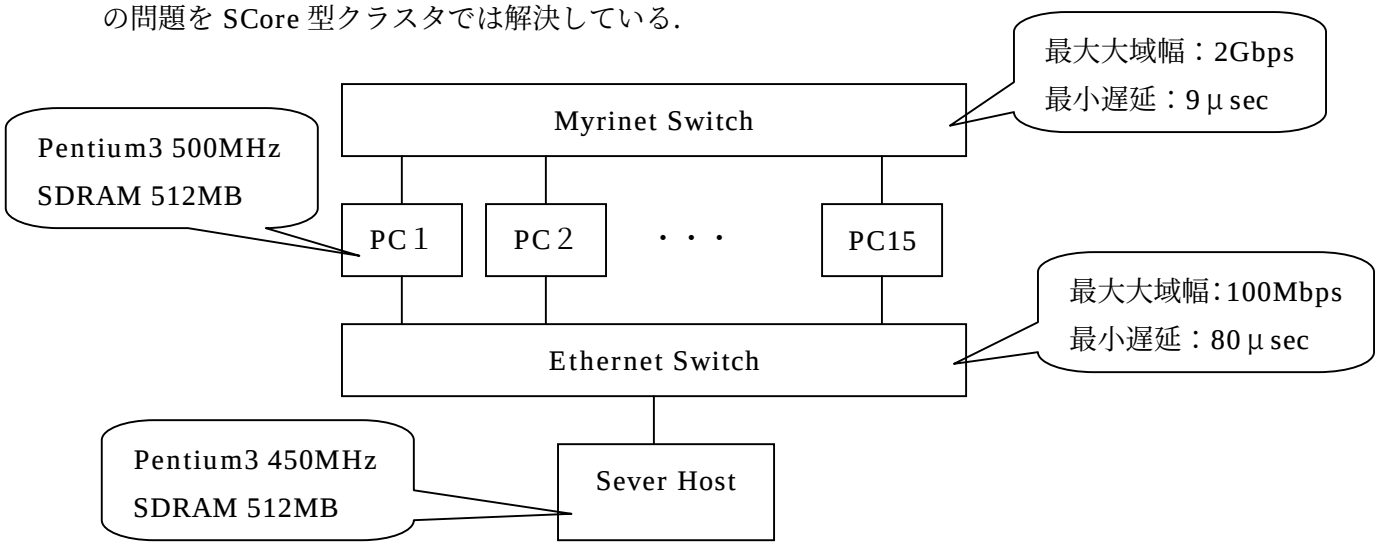


図 4 研究室のクラスタの構成

３．データマイニング

3.1 データマイニングの概要

現在の情報化社会には情報が欠かせないものになっている.そのためにいろいろなところで情報収集を行い情報量は膨大なものとなっている.収集した情報を基にデータをとろうとしても膨大な量の情報から有益なものを見つける事は容易ではない.データマイニングとは,そのような大量の情報から有益なルール,パターン,相関関係等を導く事である.有益なものとはその情報を基に実行可能であることが条件である.

3.2 データマイニングの流れ

データマイニングは次のような主に３つの主要な段階からなる(図４).

最初の段階は,データの準備であり大量のデータから必要なデータの抽出などの前処理を行う.このことにより大量のデータから余分なものを取り除いたデータができる.

次の段階はデータに隠れた価値のある情報を把握しやすくするために圧縮,変換処理を行う.大量のデータを扱うためもとのデータの形式のままであると処理が重くなってしまう.そのため分析しやすくするために前の段階でててきたデータを簡易化し,次の処理を行いやすいように変換する.最初の段階と次の段階で,データの前処理を行う.

最後の段階はデータの分析である.この処理によりいろいろな情報,即ちルールやパターンなどを見つけ出す.しかし,ここで出てくる情報すべてが有益であるわけではない.そのため出てきた情報をさらに吟味し,有益でその情報により実行可能かどうかを判断する.ここでデータマイニング処理より収集した情報を用いて戦略的意思決定が行われる.

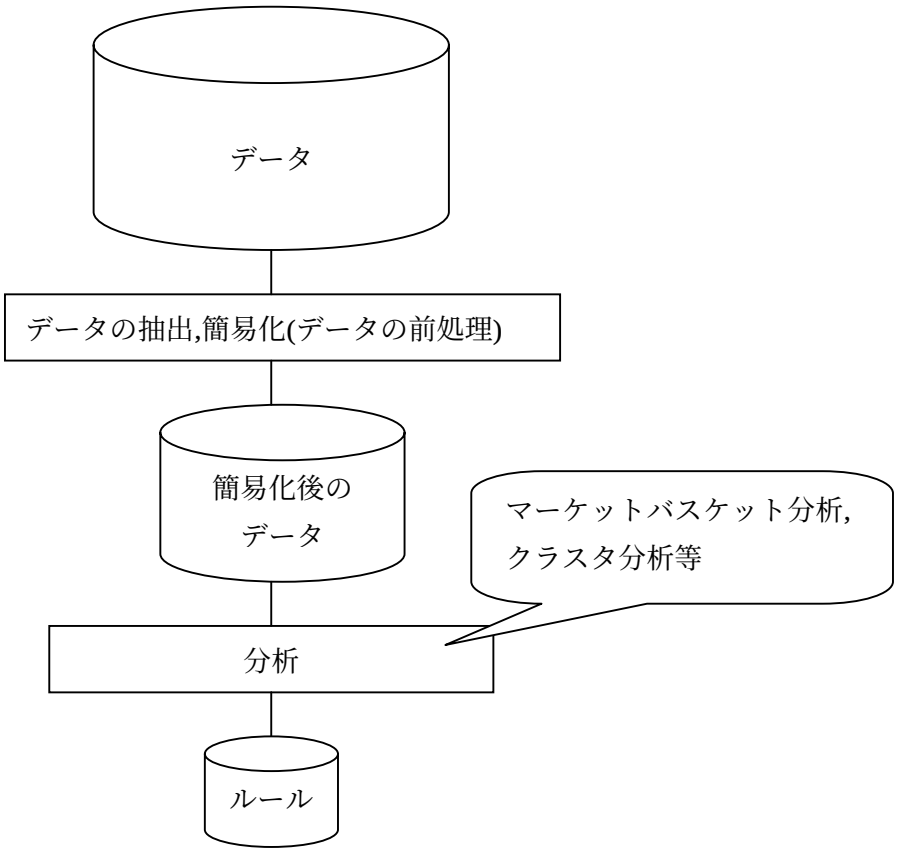


図 5：データマイニングの流れ

3.3 データマイニングの手法

3.3.1 マーケットバスケット分析

マーケットバスケット分析は,買い物かごや取引レコードでアイテムが一緒に買われるグループを見つけだすのに使われるデータマイニングの１つの手法であり,小売業の分析と密接に結びついている.イメージとしては誰かがスーパーマーケットで買ったさまざまな商品の入ったショッピングカートを思い浮かべれば分かりやすい(図５).買い物かごには食品や生活用品など,いろいろな商品の組み合わせが入っており,1 人の消費者が 1 回の買い物で購買したものを示している.ショッピングカート１つにつき１人の顧客の買い物データとなり,ショッピングカート毎にさまざまな商品の組み合わせが存在し量や時間もそれぞれ違うものになる.マーケットバスケット分析では,顧客が”何を”買ったという情報から,”なぜ”特定の購買をするのかという洞察を得る.さらにそこからさまざまな商品の同時購買されやすさが得られ,ルールとして表現できる.ルールの例としてはマーケットバスケット分析の結果の典型的な事例としては,『ツーバイフォーの板の購買者は同時にくぎを購入する』,『ペンキを買った顧客はハケも購入する(しかし逆はない)』,などがある.

手法として,特定の買い方で何が買われるかや,どのアイテムが一緒に買われるかを知り

たい時にマーケットバスケット分析は役に立つ.

分析より得られた情報は,

- 店舗内のレイアウト計画
- 特売商品を同時購買されやすい商品郡は１つだけに限定し同時購買をねらう
- 製品のバンドル販売
- 片方の商品だけしか買わなかった場合に地方の商品のクーポンを提供する

などさまざまな目的で活用できる.以上は顧客を識別しない場合であるが,取引が匿名でない場合には,顧客毎の時間の要素も入れて顧客毎の購買履歴を対象としたマーケットバスケット分析ができる[4].

図 6：マーケットバスケット分析

3.3.2 クラスタ分析

クラスタ分析は,互いに似ているレコードを見いだすようなモデルを構築する手法である(図６).例えば人間が複雑な問題を理解しようとする場合,自然な傾向として,対象をより小さく分類し,もっと単純に説明できるように工夫する.図 6 は温度と光度を軸にとった星についての散布図である.これらを分類するために距離や重心などを用いて類似しているものを見つけグループを形成する.

類似しているこのグループはクラスタと呼ばれる.このように大きな情報からいろいろな情報を手に入れようとする時にまず小さなクラスタを形成しそこから分析していくことをクラスタ分析という.

分析目標は未知のデータの類似性を発見することなので,この手法は本来的に探索的データマイニングである.また,クラスタごとに分類済みのデータは,どんな分析を始める際にも大変良いデータとなる.そのため,データに何があるかを知り,どのようにデータを利用すればよいかを理解するスタート地点を,クラスタは提供できるのである[4].

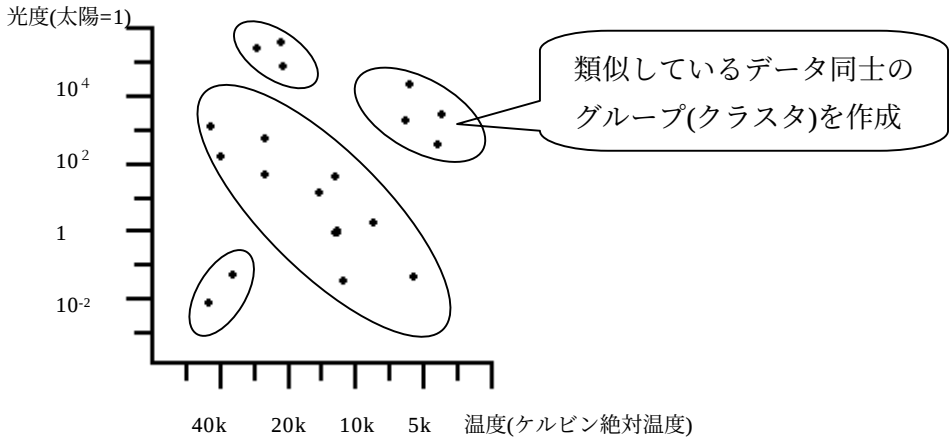


図 7：温度と光度による星のクラスタ図

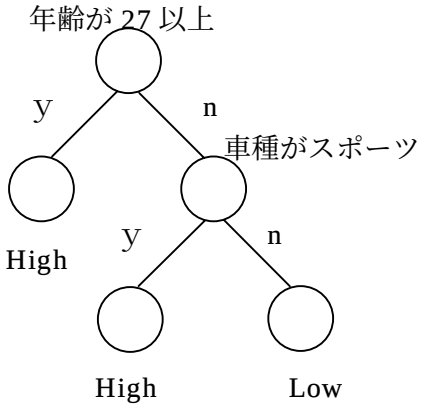
3.3.3 決定木

決定木は,データマイニングで用いられる classification アルゴリズムの一手法である.他の classification アルゴリズムと比べて高速であり,同程度以上の質の解が得られるという利点がある.

決定木では,トレーニングセットと呼ばれるデータ集合が与えられる(図 1(a)).1 つのデータ(レコードと呼ぶ)は,複数の属性と 1 つのクラスを持つ.属性がそのレコードを分類するための情報であり,クラスは分類先の情報である.属性は,カテゴリ値と呼ばれる離散値を取る場合(図 1(a)の Car Type 属性)と連続値をとる場合(図 1(a)の Age 属性)がある.決定木生成アルゴリズムにより,トレーニングセットから決定木(図 1(b))と呼ばれる分類木が生成される.これは,クラスの与えられていないテストセットと呼ばれるデータを,属性の値に応じて適切なクラスに分類するための木である[11].

年齢	車種	リスク
23	ファミリー	High
17	スポーツ	High
43	スポーツ	High
68	ファミリー	Low
32	トラック	Low
20	ファミリー	High

(a)



(b)

図 8：トレーニングセットと決定木

４．データマイニングの並列化

4.1 問題定義

研究室の HP へのアクセスはいろいろなところから毎日ある.そのアクセスログは日々増加しているため,膨大な量になっている.このアクセスログを使い,マイニングすることにより法則性を導き出す.しかしこの処理は,膨大な量のデータを扱うため処理時間がかなりかかってしまう.そこで PC クラスタ上で並列処理を行うことにより,データマイニングの処理時間の短縮を狙う.

今回作成したプログラムでは前章で述べたデータマイニングの流れの中の,必要なデータの抽出,抽出したデータの圧縮,変換処理を行い簡易化する部分を作成した.さらにその簡単化した段階での情報の統計を行った.

アクセスログは次のような形式になっている

＜アクセスログの例＞

133.19.36.152 - - [28/Jun/2002:18:17:02 +0900]
"GET /apache_pb.gif HTTP/1.1" 200 2326
"http://www.hpc.cs.ritsumei.ac.jp/index.html.ja.jis"
"Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"

133.19.36.152 ： HP **にアクセスしてきた** IP
28/Jun/2002:18:17:02 +0900 ： HP **にアクセスしてきた時間**
GET /apache_pb.gif HTTP/1.1 ： **メソッド(データ転送の** URL **を含んでいる.この** URL **をデータとする)**
200 ： **応答コード(正常)**
2326 ： **送信したバイト数**
http://www.hpc.cs.ritsumei.ac.jp/index.html.ja.jis ： **リファラー(ジャンプ前の** URL)
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0) ： **ユーザエージェント(ブラウザ**,OS **などの情報を持っている**.)

データの抽出ではアクセスログから次のものを抜き出している.

IP　Time　referrer　URL

これらのデータ同士の相関をとることで次のような情報がわかる.

- ページ間の関係(URL,referrer の関係よりページのジャンプもととジャンプ先)
- ある時期でのアクセス情報(Time と他の情報)
- ページのクラスタ(IP,referrer によりユーザグループがどのような所を見るか)

・ おおまかなユーザーなどの情報等(IP と URL の関係によりどのようなページを見るか)

4.2 アルゴリズム

今回作成した処理の抽出部,簡単化部,統計部のアルゴリズムを次に示す(図８).

(1) アクセスログから必要な情報を抽出する

ログからデータを取り出しファイルに保存する.取り出すファイルは次の 4 つである.

・IP,Time,URL,referrer をまとめて抜き出したもの

・IP

・URL

・referrer

IP,Time,URL,referrer をまとめて抜き出したものは今後のルール作成のためのデータとして使うためにファイルに出力される.抜き出したデータは表２のような形式で保存される.

IP,URL,referrer のそれぞれのデータは別々のファイル(IP,URL,referrer)に格納されソートし重複をとる.このファイルはデータの簡単化を行うの時に使用する.

(2) データの簡単化

データの簡単化は(1)で出てきたファイルを使い行う.IP,URL,referrer はログから必要な情報を抜き出したものと,重複をなくしたものと比較し簡単化される.Time は 2001 年 1 月１日 0 時 0 分からの経過した秒数で表す.データはすべて数値化される.簡単化したデータは表 2 のような形式になる.

表 2 データの保存形式

	抽出したデータ	簡易化後のデータ
IP	133.19.36.152	1772
Time	28/Jun/2002:18:17:02 +0900	46981101
URL	GET /apache_pb.gif HTTP/1.1	1899
referrer	http://www.hpc.cs.ritsumei.ac.jp/index.html.ja.jis	2997

すべて簡単化することにより,今後の処理での情報の把握が簡単に行える.

(3) 統計

統計は２つのデータから二次元で行う.二次元にとったデータは大きすぎるので見て分かるように出力するのは困難である.そこで多く現れている部分のみ注目してデータを出力する.

.

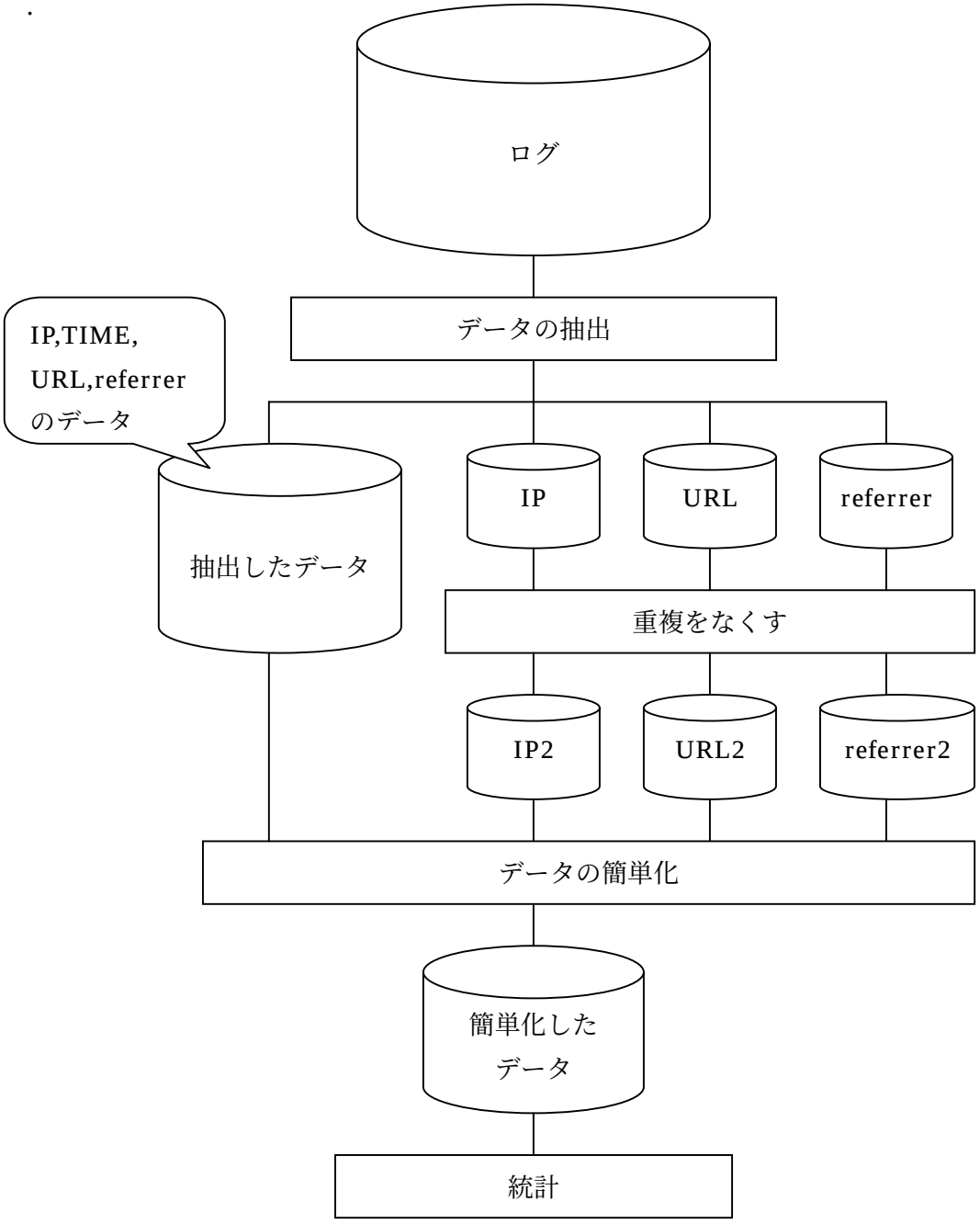


図 9：アルゴリズム

4.3 並列化手法

4.3.1 データ抽出

データの抽出の並列化は図 9 に示すようにアクセスログからの抽出を複数のクラスタ上で分担して行い,出力も別々に行う.アクセスログの分割は,クラスタ毎にファイルの読み始める場所をずらす事で行う.アクセスログ出力を別々に行うことでこの後の処理の並列化を行いやすくする.抽出はログから繰り返し行っているので並列化するのには適している.

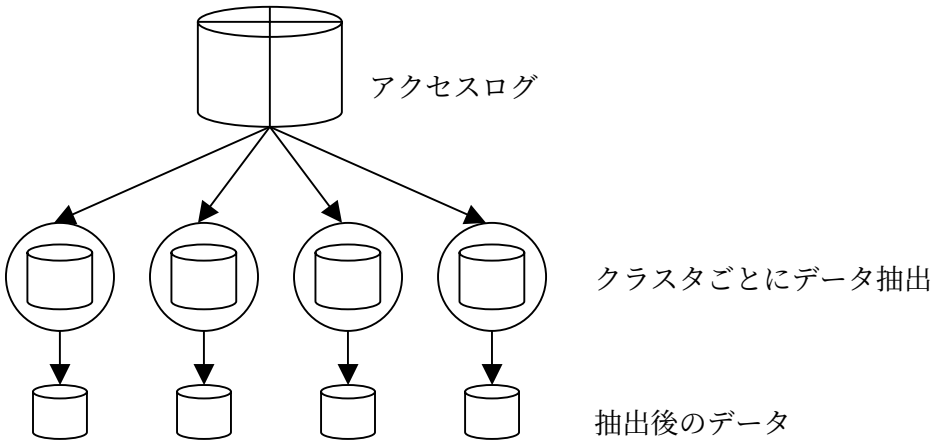


図 10 データ抽出の並列化

4.3.2 データの簡易化

データの簡易化の並列化は図 10 に示すように,データの抽出により作成したファイルをそれぞれクラスタに割り当て処理する.この場合も出力は別々に行う.ファイルを分割してそれぞれのクラスタ上で行っているので,並列処理部分が全体であり,理想的な分割ができている.

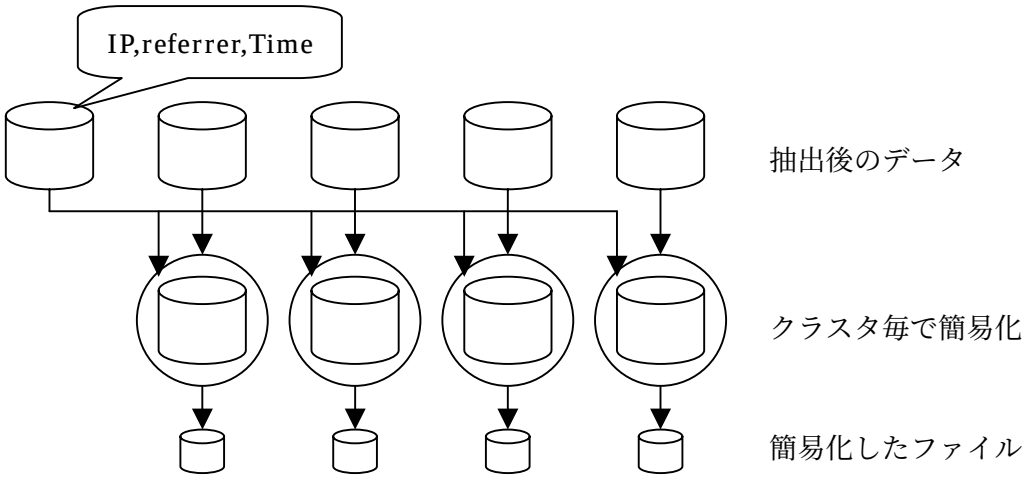


図 11 データの簡易化の並列化

4.3.3 データの統計

データの統計の並列化は図 11 に示すように簡易化したデータをそれぞれのクラスタ上で統計をとった後に， 1 つの統計結果のファイルにまとめる.この処理も統計結果をまとめる部分まではそれぞれ独立で並列に行っているためほぼ理想的な分割ができている.そのため速度向上も期待できる.

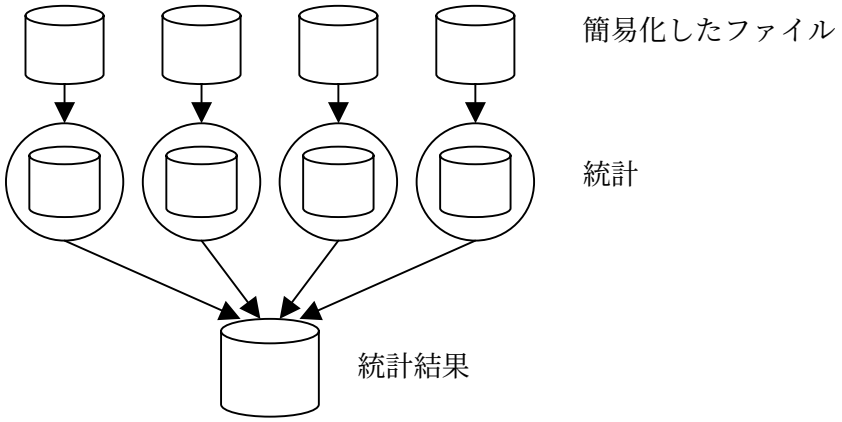


図 12 データの統計の並列化

4.4 実行結果

アクセスログからのデータの抽出,簡易化処理により作成されたファイルは表３のようになった.

表 3 出力データ

ファイル	レコード数	ファイルサイズ[byte]
アクセスログ	114597	20.2M
IP	8394	117k
URL	6864	256k
referrer	6860	752k
簡易化したデータ	114537	2.56M

種類数：ファイルの中にでてくるデータの種類数

表２を見るとアクセスログと簡易化したデータの種類数が異なっている.簡易化はアクセスログのデータをそれぞれ簡易化しているものなので種類数は同じになるはずである.これはアクセスログの中に簡易化する処理の中での予期せぬ場合のデータがあった場合

にエラーとしてファイルに出力しないために起こった.

簡易化したデータから統計をとり IP.URL,referrer に注目して見ると次のようなことが分かった.

IP：HPC 研究室のマシンからの HP アクセスが多い(特に 10 月)

URL：特定の卒論ファイルの DL が多い(特に 7 月)

referrer：他の HP からジャンプしてくるのではなく自分で URL を打ち込んで HP にアクセス,もしくはお気に入りからアクセスする人が多い.

referrer と URL に注目してみるとページ間の関係がわかる.表４は Referrer と URL の統計結果から頻度の多いものを抜き出したものである.

表 4 Referrer と URL の統計結果(頻度の高いもの抜粋)

URL	referrer
GET/Gazou.html/HTTP/1.1	-
GET/HTTP/1.0	-
GET/rlinks.htm/HTTP/1.0	-
GET/members.htmlHTTP/1.0	http://www.hpc.cs.ritsumei.ac.jp/
GET/ Gif や jpg,cgi 等	http://www.hpc.cs.ritsumei.ac.jp/

Gif や jpg,cgi などがかなりあったが,これはあたりまえなので統計ではほとんど省く.

統計をとることでログからは次のようなことが分かった.

- referrer がない状態(自分で URL を打ち込んで HP にアクセス,もしくはお気に入りからアクセスする)が多く,特に HPC 研究室 HP の index ページを読み出すものが多い.
- referrer がない状態を抜いて統計をとると研究室の HP からのアクセスが多い事がわかる.

IP と referrer に注目することでユーザがどのような場所から来るかが分かる.つぎの表 5 は IP と referrer からの統計結果の頻度の高いものを抜き出したものである.

表 5 IP と referrer の統計結果(頻度の高いもの抜粋)

IP	referrer
203.139.121.210：J 企業	-
202.212.5.34~39：N 企業	-
61.121.102.67：A 企業	http://www.hpc.cs.ritsumei.ac.jp/papers
61.205.123.246：C マンション	http://www.hpc.cs.ritsumei.ac.jp/
61.78.61.162~168：AU 国	-

- 多くの企業からのアクセスが多く,その中でも研究室の卒論のページにアクセスしている所が多い.その事から,本研究室の研究内容が企業での仕事に関係し,その情報が必要とされていることがわかる.
- Cマンションはあるマンション全体でのIPであり,学生からの利用も多い事がわかる.
- 表にはあまり書いていないが AU 国等,外国からのアクセスも多くあった.

IP と URL に注目することでユーザがどのような場所を見ようとしているのか(目的)がわかる. つぎの表 6 は IP と URL からの統計結果の頻度の高いものを抜き出したものである.

表 6 IP と URL の統計結果(頻度の高いもの抜粋)

IP	URL
165.93.176.226 ：N 大学	GET/papers/ikeda.pdf/HTTP/1.0
131.113.69.36 ：K 大学	GET/pvm/jpvm-020528.pdfHTTP/1.0
210.140.252.253：M 企業	GET/pvm/indexHTTP/1.0
61.205.123.246 ：C マンション	GET/~furukawa/HTTP/1.0
61.205.123.246 ：C マンション	GET/~furukawa/limit/index.htm/HTTP/1.0

ログからは以下のことが読み取れた.

- 大学や企業からのアクセスが多く,特定の研究員の卒論内容や pvm についての内容が必要とされている事がわかる.
- C マンションからのアクセスは本研究室の研究員が研究室に関する仕事をする場合に多くアクセスしたため頻度が高くなった.

５．考察

アクセスログからの抽出,簡易化を行ったあとのデータを見るとかなりのデータの圧縮がされたことが分かる.この処理により文字列だったデータが数値化されるのでデータの情報が読み取りやすく把握しやすいものとなるためその後の処理の向上が得られることが分かる.

簡易化したデータの統計結果はわかりやすいものもあるが,なぜそうなっているか理解しにくいものもあった.特に IP だけに注目してみた場合の結果と,2 つの情報の相関をとった場合(IP と referrer 等)の結果が同じようなものではなかったことが理解しにくい.普通に考えればある IP が多く出現している場合に,その IP とほかの情報との相関をとれば,多く出現している IP に関する情報も多くでてくるはずである.しかし今回はそのようなことばかりが見られるわけではなかった.理由としては,統計結果の全てを見て情報を得ているわけではないためだと考えられる.データの相関の統計結果は 2 次元のデータとして扱われるが種類数が多い.そのため,データは少なくとも約 50 万個(約 7000×7000 種類)くらいでてきてしまう.それらをすべて表示して結果を見ることは困難であるため,結果は比較的統計結果の多いものだけを抜き出して表示することで統計結果の情報を得た.つまり統計結果より得られた情報は全てのデータを見た場合のものではなく,データの中の一部にしか注目していないためそのような結果がでたのだと考えられる.このことから膨大なデータから人の目だけで有益な情報を得ようとすることは大変困難であることが分かる.少ないデータであればデータのほとんどを把握できるためいろいろな情報が見つけやすい.しかし,データが大きければ大きいほど把握できない部分が増えその部分を分析できないためにいろいろな情報が得られない場合も出てくる.このことから膨大なデータを使用する場合に,データマイニングは効率的に全てのデータを使用し処理を行うものであるため,分析結果に信頼性があることがわかる.現代の情報化社会ではデータマイニングはなくてはならないものとなっていると感じた.

６．おわりに

今回データマイニングの分析部分の作成を行った.今後データマイニングプログラムの完成とその並列化を行うという２つの課題が残る.データマイニングの分析部分はマーケットバスケット分析や,クラスタ分析などいろいろな手法がある.今回作成したデータの抽出,簡易化の結果はそれらの手法に対応するものであるため色々な手法を作成し,試してみたいと思っている.

並列化においては,前処理部分の並列化は検討により処理の向上が得られそうである.しかし,分析部分についてはまだ作成していないため,どのように並列化すればいいか分からない.並列化はその処理にあった並列方法を選ぶことでかなりの速度向上が得られる.データマイニングの分析部分にあった並列方法を選び理想的に近い速度向上を得たい.

さらにそれらの結果は研究室の HP の情報であるため,今後の HP のあり方について検討し,研究室外の人たちに立命館高性能計算研究室のことをより知ってもらえるものとなるよう工夫していきたい

謝辞

本研究の機会を与えてくださり、貴重な助言、ご指導いただきました山崎勝弘教授、小柳滋教授に深く感謝いたします.また本研究にあたり、また、いろいろな面で貴重なご意見や、励ましを頂きました本研究室の皆様に深く感謝いたします.

参考文献

[1] 湯浅　太一、安村　通晃、中田　登志之：“bit 別冊　はじめての並列プログラミング”、共立出版、1998.

[2] 大森　健児：“並列プログラムの基礎”、丸善株式会社、1990.

[3] Rohit Chandra, Leonardo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald, Ramesh Menon：“Parallel Programming in OpenMP”、Morgan Kaufmann Publishers、2001

[4] マイケル J.A.ベリー、ゴードン・リノフ 著　SAS インスティチュート ジャパン、江原　淳、佐藤　栄作 共訳：“データマイニング手法”、海文堂出版、1999

[5] IBM Corp.ジョセフ･P･ビーガス 著 社会調査研究所、日本 IBM 共訳：“ニューラル・ネットワークによるデータマイニング”、日経 BP 社、1997

[6] 石川　裕、佐藤　三久、堀　敦史、住元　真司、原田　浩、高橋　俊行：“Linux で並列処理をしよう”、共立出版、2002

[7] 奥村　晴彦：“C 言語による最新アルゴリズム辞典”、技術評論社、1991.

[8] 小柳　滋、久保田　和人、仲瀬　明彦、酒井　浩：決定木の並列化とその評価、1999

[9] 米田　健治：事例ベース並列プログラミングの実験と評価（Ⅱ）、立命館大学理工学部情報学科卒業論文、1998

[10] 内田　大介：OpenMP による並列プログラミング[Ⅰ]、立命館大学理工学部情報学科卒業論文、2000.

[11] 土屋　悠輝：OpenMP による並列プログラミング[Ⅱ]、立命館大学理工学部情報学科卒業論文、2000.

[12] OpenMP Home Page：http://www.openmp.org.

[13] PC Cluster Consortium：http://www.pccluster.org/.

[14] PC クラスタ超入門 2000：http://mikilab.doshisha.ac.jp/dia/smpp/cluster2000/