配列データに対する頻出パターンの並列抽出方式

周藤 俊秀　田村 慶一　森 康真　北上 始

広島市立大学情報科学部 〒731-3194 広島県広島市安佐南区大塚東 3-4-1

E-mail: {tosihide,ktamura,mori,kitakami}@db.its.hiroshima-cu.ac.jp

あらまし 本論文では，大規模な配列データベースから頻出パターンを高速に抽出するために，ネットワークで接続された複数台の計算機を用いて，既に著者らによって開発されている Modified PrefixSpan 法を並列化する方式について提案している．動的な負荷分散機構をもつ本方式では，ソケット通信と MPI ライブラリを用いて，複数台の計算機間の通信を行っている．また，マスタプロセスが複数のスレーブプロセスと送受信するために 1 つのマスタプロセスの中にマルチスレッドを用意している．マスタプロセスでは，初期段階で生成された部分木の集合を管理するグローバルジョブプールとスレーブプロセスの制御を行っている．実験結果として，8 台で 1 台に比べて 6 倍程度の性能向上が得られた．今後，動的な負荷分散処理を強化し，その並列性能を評価する予定である．

キーワード 並列 DB，データマイニング，配列データ，動的負荷分散，PC クラスタ

Parallel Extraction Method of Frequent Patterns in Sequences Database

Toshihide SUTOU　Keiichi Tamura　Yasuma Mori　and　Hajime Kitakami

Faculty of Information Sciences, Hiroshima City University　3-4-1 Ozukahigashi, Asaminami-ku Hiroshima-shi, Hiroshima, 731-3194 Japan

E-mail: {tosihide,ktamura,mori,kitakami}@db.its.hiroshima-cu.ac.jp

Abstract In order to extract efficiently the set of frequent patterns from large-scale sequence databases, this paper proposes the method of parallelization for Modified PrefixSpan method, developed by authors, using multiple computers connected to Internet. The method with a dynamic load balancing mechanism is achieved communications among multiple computers using socket and MPI library. Moreover, it includes multi-threads to achieve communications between a master process and multiple slave processes. The master process controls both a global job pool to manage the set of subtrees generated in an initial processing and multiple slave processes. As a result, 8 computers were roughly 6 times faster than 1 computer in our experiments with trial implementation. In the feature, we will implement the method of enhanced dynamic load balancing and evaluate the performance of it.

Keyword Parallel DB, Data Mining, Sequences Data, Dynamic Load Balancing, PC Cluster

1. はじめに

モチーフは，アミノ酸配列上における特徴的なパターンであり，生物の進化の過程で保存されてきた蛋白質の機能に関係していると考えられている．アミノ酸配列は 20 種類のアルファベットから構成され，特徴的なパターンを抽出することは文字配列上の頻出パターンを取り出すことになる．大規模なアミノ酸配列からのモチーフ発見のための高速な頻出パターン抽出アルゴリズムの研究が重要となっている．

文字配列上のさまざまな位置にある頻出パターンを見つけることができる PrefixSpan 法[1]が提案されている．PrefixSpan 法を直接アミノ酸配列の特徴的なパターン抽出に利用すると余分なパターンが数多く抽出されるため，計算処理時間が大変かかってしまうという課題があった．我々はこの問題を解決するために，PrefixSpan 法にワイルドカード数を制限する仕組みを導入した Modified PrefixSpan 法[2][3]を提案している．アミノ酸配列でモチーフとなり得る特徴的なパターンはワイルドカード数が固定である可能性が高い．ワイルドカード巣を設定することにより抽出処理の枝刈を行うことができる．Modified PrefixSpan 法は，800 本程度のアミノ酸配列データ(25～4200 文字)に対して，計算時間の短縮や余分なパターンを削除することに成功している．

本論文では，大規模な配列データベースから頻出パターンを高速に抽出するために，ネットワークで接続された複数台の計算機を用いて， Modified PrefixSpan 法を並列化する方式について提案する．並列化の特徴は動的な負荷分散を実現するために Modified PrefixSpan 法による頻出パターン抽出処理を，ある一

定の文字までの頻出パターン抽出を行い，それ以降に続く頻出パターン抽出を複数の計算機に分けることを行ったことである．以後，この一定の文字までの頻出パターンに続く頻出パターンを部分頻出パターンとする．

並列処理を管理する１つの計算機が複数の部分頻出パターン抽出処理に全体の抽出処理を分割する．分割した仕事を，１つ１つ並列処理を行う計算機に配る．部分頻出パターン抽出を終えた計算機は，次の部分頻出パターン抽出処理の仕事を受け取り，部分頻出パターン抽出の仕事がなくなるまで処理を続けていく．Modified PrefixSpan 法の特徴として複数に分けた部分頻出パターン抽出処理の負荷はそれぞれ異なる．早く部分頻出パターンの抽出を終えた計算機は，次の抽出処理を行う．ある計算機が負荷の重い処理を処理している間に他の計算機が次々に処理を続行することで動的な負荷分散を行う．

部分頻出パターン抽出を基とした Modified PrefixSpan 法を実際の PC クラスタ上に実装を行った．実験結果として，8 台で 1 台に比べて 6 倍程度の性能向上が得られた．検証実験ではユーザの試行による閾値の設定を行った結果有効な並列効果を得られることを確認した．自動的に最適な負荷分散を行うために，自動的な閾値の設定法と極端に負荷の重い部分パターン抽出処理を扱うための負荷のグローバルな管理法の実装を今後行う予定である．また，さまざまな特徴を持った配列データを用いて並列性能を評価する予定である．

本論文の構成は以下の通りである．２章では本研究に関連する研究を紹介する．３章では Modified PrefixSpan 法の説明をし，４章ではその並列処理の詳細設計について述べる．５章では検証実験の結果を述べ，６章で動的負荷分散方式の検討を行っていることを報告し，７章でまとめる．

2. 頻出パターンの抽出法についての関連研究

マルチプルアライメントを用いて，あるアルファベット集合Σの上で定義されている N 本の配列集合 $\{S_1,S_2,\ldots S_N\}$の平均長を L とすると，その配列集合からモチーフを見つけ出す方法は，時間計算量が $O(L^N)$ となるので膨大な計算時間を要する．それにもかかわらず,極めて限定された頻出パターンしか得られない．

これに対して，Timothy らが開発した MEME[4]というシステムでは，アライメントされていない N本の配列集合からおおまかな情報としてモチーフの長さ Wと各配列中の位置を利用者に指定させ，その部分に対してマルチプルアライメントを行った後，統計的手法である期待値最大化アルゴリズムをM回繰り返すことにより，複数の頻出パターンを抽出している．これにより，MEME では，時間計算量を $O((NM)^2W)$ に抑えているが，抽出される頻出パターン中にワイルドカードを含まないため，PROSITE や Pfam[5]などで見られるモチーフとは直接結びつきにくい．

しかし,Jonassen らが開発した Pratt というシステム[6]では，利用者に抽出パターン P の最大長 Wを指定させ，各配列から長さ Wの部分配列（以後，セグメントと呼ぶ）を全てとりだし，それらの集まりとして定義されるセグメント集合 B_Wからワイルドカードを含む頻出パターン P を全て抽出することができる．ただし，配列の終端付近からも頻出パターンを抽出するために,長さ Wの部分配列を取り出す前に各配列の後ろに W-1 個のダミー記号（Σには存在しない）が予め追加されている．全ての頻出パターンは k 文字長のパターン集合から k+1文字長のパターン集合を再帰的に構成する方法により抽出される．この処理は，最大長が W文字になるまで繰り返し計算が行われる.計算量の見積もりは難しいが，効率的な抽出のために，セグメント集合 B_Wは各構成要素 a が i 番目にもつセグメントの集合 $b_{i,a}(\subseteq B_W)$の集まりとして構造化されている．例えば，パターン P から $P'=P\text{-}x(j)\text{-}a$ を構成する場合，P' は P が前方一致するセグメント集合 M_p と要素 a が $L(P)+j+1$ 番目に位置するセグメント集合 $b_{L(P)+j+1,a}$ との共通セグメントの全てと前方一致する（ただし，j は 0 以上の整数だが，$L(P)+j+1 \leqq W$を満たす）．これにより，P'が存在する配列数を数え，その数が利用者によって与えられた最小支持数 N_{min} 以上であれば,P'を頻出パターンとしている.ただし,$L(P)$ はワイルドカード領域を含むパターンPの長さを表す.

これに対して著者らが開発した Modified PrefixSpan 法は，抽出される頻出パターン P の最大長 Wの利用者による指定が不要である．その代りに，利用者にワイルドカード領域の最大長 Vを指定させている．これにより，著者らの方法では，利用者に予期していなかった長さの頻出パターンを発見する可能性を与えている．また，この方法も，Pratt のように，長さが kの頻出パターンから長さが k+1の頻出パターンを再帰的に作成しながら全ての頻出パターンを抽出しているが，その処理過程においては，頻出パターン長の制限につながるセグメント集合を作らずに，直接 N本の配列集合から頻出パターンを抽出している．効率的な処理を行うために，N 本から成る配列集合 $\{S_1,S_2,\ldots S_N\}$の構造化は，配列の構成要素 a と配列 S_i の組み毎に,配列 S_iの先頭から何番目に a が存在するかを示す位置情報の集合 $T_{a,j}$を作成することにより達成している．例えば，パターン P から $P'=P\text{-}x(j)\text{-}a$ かつ $0 \leqq j \leqq V$ を構成する場合，各配列 S_i において

*last(P,i)*を P の最後尾にある要素の位置情報とすると，位置情報の集合 $T_{a,j}$中に *last(P,i)+1*から *last(P,i)+V+1* までの区間に当該要素 a の位置情報があるかどうか調べ，その位置情報が存在する配列 S_iの数が最小支持数 N_{min}以上であれば P'を頻出パターンとしている．

3. Modified PrefixSpan 法

PrefixSpan 法は事前に支持率(頻出パターンの出現割合) を決める．Modified PrefixSpan 法はこれにワイルドカード数という，文字と文字の間の任意の文字数も事前に決める．例えば，A***B と A****B (* はどの文字でもよいことを示す)は，PrefixSpan 法では AB というパターンとみなすが，Modified PrefixSpan 法では文字間のワイルドカード数を指定することで文字間のワイルドカード数が異なるパターンは同じパターンとはみなさない．

アミノ酸配列でモチーフとなり得る特徴的なパターンはワイルドカード数が同数のものである．つまり，A***B と A****B は A3B，A4B と区別している．ワイルドカードの導入によりモチーフとなる可能性の低いパターンの抽出を以降行わないことで枝刈を行っている．

PrefixSpan 法は現れる頻出パターンから配列の最後の位置までを頻出パターン抽出の対象とするのに対し，Modified PrefixSpan 法は現れる頻出パターンからワイルドカード数＋ 1 の先までの長さの文字列を頻出パターン抽出の対象とするため，配列データが長くなるほど Modified PrefixSpan 法より高速化が期待できる．

Modified PrefixSpan 法は，与えられた文字列データの集合に対してまず，長さ 1 の頻出パターンを求める．次に求めた長さ 1 の頻出パターンをそれぞれ 1 文字目とし，長さ 2 以上の頻出パターンを求める．例として表 1 に示す二つの配列に対して，支持率を 100，ワイルドカード数を 3 とした場合に現れる頻出パターンを図 1 に示す．

Modified PrefixSpan 法は木構造で表すことができる．図 1 では，まず支持率を満たした長さ 1 の頻出パターンを抽出する．図 1 では省略されているが 2 本の配列に共通する「K，L，M，N，P，R，S，T」 が頻出パターンとみなされる．次に各文字を接頭辞とし，2 文字目以降の頻出パターンの抽出を行う．図 1 に示されるように，例えば 1 文字目が M の場合，2 文字目以降に現れる頻出パターンを N，S，SP としたときに，すべて文字と文字の間のワイルドカード数が等しくないために頻出パターンとはみなされていない．しかし，1 文字目が K の場合，2 文字目以降に現れる頻出パターンを，L，LR としたときに，ワイルドカード数が等しくなっているので，この場合は頻出パターンとみなす．このように，Modified PrefixSpan 法は，次々に枝分かれをして頻出パターンの抽出が行われている．

表 1　配列データベース

配列 ID	配列
1	MFKALRTIPVILNMNKDSKLCPN
2	MSPNPTNHTGKTLR

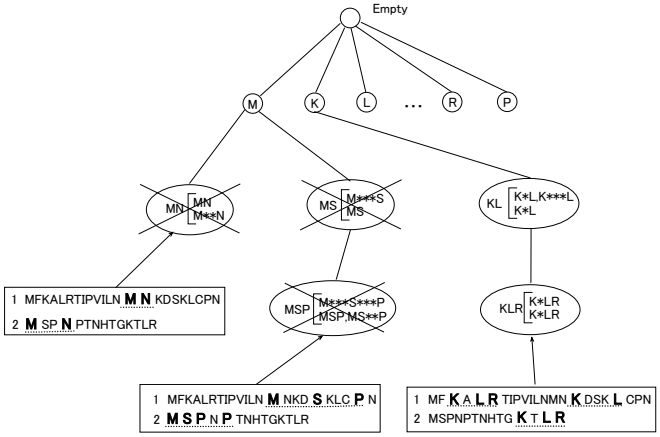

図 1　Modified PrefixSpan 法の頻出パターン抽出

4. 並列 Modified PrefixSpan 法

4.1. 部分頻出パターン抽出

Modified PrefixSpan 法の並列化では，任意の深さまでの頻出パターンの抽出を行い，それによって得られる複数の部分頻出パターンの処理を各プロセスが受け取り，探索を続けるという手法を用いる．以下，探索をする任意の深さのことを閾値とする．Modified PrefixSpan 法は 1 文字目のパターンをルートノードの子ノードとする木構造の探索の処理とみなすことができ，容易に複数の部分頻出パターン抽出処理に分け，部分頻出パターン抽出処理は他の抽出処理とは独立して処理を行うことができる．

図 2 に閾値を 2 とし，プロセス数を 4 とした場合を図示する．なお，以降では，複数の部分頻出パターン抽出処理(各プロセッサが探索する部分木)をジョブと呼ぶ．

アミノ酸は，アルファベットの記号「A，C，D，E，F，G，H，I，K，L，M，N，P，Q，R，S，T，V，W，Y」に相当し，「B，J，O，U，X，Z」を除く 20 個のアルファベットで使用される．

図 2 の場合，閾値が 2 なので，アミノ酸のアルファベットの組み合わせが最大で 20 文字×20 文字の 400 個のジョブが現れる．このジョブを例えば 4 台の

計算機で抽出すると,各計算機は平均で 100 個のジョブを行うことになる.

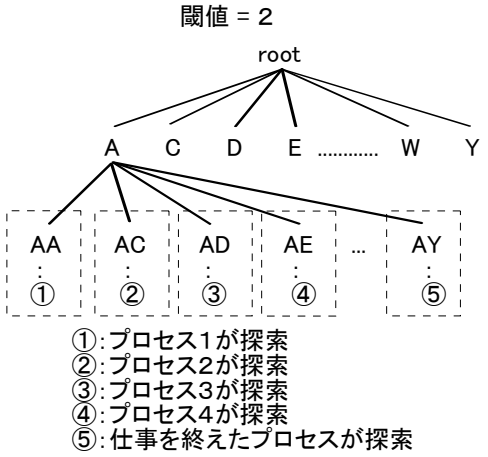

図 2 閾値 2 の場合の並列木探索

4.2. システムの詳細設計

ジョブを生成し管理するプロセスをマスタプロセスとし，Modified PrefixSpan 法による頻出パターンの抽出処理をおこなうプロセスをスレーブプロセスとした.生成したジョブは Global Job プールと呼ばれるジョブを格納するデータ構造に挿入する．マスタプロセスと各スレーブプロセスとのデータの送受信にはソケット通信と MPI ライブラリを利用した.

ジョブを動的に管理するためにマルチスレッドを利用した．よってマスタプロセスとスレーブプロセスとの通信とマスタプロセスからのジョブの取り出しは並列に行われる.

並列 Modified PrefixSpan 法の処理フローを図 3 に示す.

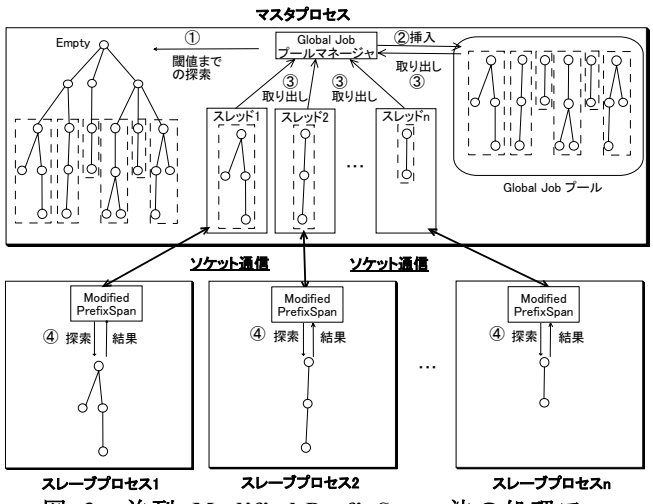

図 3 並列 Modified PrefixSpan 法の処理フロー

図 3 の処理のステップは以下の順でおこなわれる.

①:マスタプロセスで閾値までの頻出パターン抽出処理を行う.

②:①で得られる複数のジョブを Global Job プールに挿入する.

③:Global Job プールにあるジョブをスレーブプロセスの数だけ作り出したスレッドが，対応するスレーブプロセスにソケット通信によるデータ送信をする.

④:各スレーブプロセスは，送られてきたデータを Modified PrefixSpan 法による頻出パターンの抽出を行う.

⑤:スレッドは,スレーブプロセスの計算の終了を待ち,終了状態を確認すると②以降を繰り返す.

5. 検証実験

3 章で示した詳細設計をもとに，実際の PC クラスタ上に実装をおこない検証実験をおこなった．評価の方法として，以下の実験を行った.

(1) 性能向上比
支持率，ワイルドカード数，閾値を設定し配列の種類を変え,計算機の台数を 2 台から 8 台へと増やした場合の実行時間を計測し，その性能向上比を計算する．計算機の台数を増やすことでどの程度の台数効果が得られるかを確認する.

(2) ワイルドカード数と支持率の変化による台数効果
配列の種類と閾値を固定して支持率またはワイルドカード数を変化させ，計算機の台数を 2 台から 8 台へと増やした場合の実行時間を計測する．既存の Modified PrefixSpan 法と比較してワイルドカード数や支持率に関係なくスピードアップを図れているかを確認する.

(3) 閾値の変化
支持率，ワイルドカード数，配列の種類，計算機の台数を設定し，閾値を 1，2，3 場合の閾値までの実行時間と残りの頻出パターン抽出の実行時間を計測する．閾値を変更することでどちらがどの程度の実行時間かかるのかを確認する.

本研究で構成したクラスタマシンは CPU が PentiumIII 450MHz でメモリを 128 M 搭載したものを 8 台，Network は 100M スイッチ，OS は Redhat Linux 8.0，コンパイラは Intel C/C++ complier 6.0 for linux，MPI は MPICH(バージョン 1.2.4) を使用した.

この実験に使用したアルファベットの文字列は，アミノ酸配列のモチーフを含む配列データをデータベー

ス化している PROSITE の Kringle と Zinc Finger を使用した．詳細データを表 3 と 表 4 に示す．

表 2　使用データの詳細 1

	データ件数 (件)	総長 (byte)
Kringle	70	23385
Zinc Finger	467	245595

表 3　使用データの詳細 2

	平均長 (byte)	最大長 (byte)	最小長 (byte)
Kringle	334	3176	53
Zinc Finger	525	4036	34

本実験は検証のため，手軽に実験結果を得られる配列データを使用した．

5.1. 評価実験－1

検証実験を示す前に本実験で設定したマスタプロセスとスレーブプロセスの起動関係を示す．計算機数を 2 とした場合，1 台をマスタプロセスに，別のもう 1 台をスレーブプロセスにすると，効率のよい並列処理が行われていないことになる．なぜなら，マスタプロセスは閾値までの探索しかしておらず，残りの部分木のノードに続く頻出パターンの抽出をもう 1 台で実行することになり，これでは Modified PrefixSpan 法を 1 台の計算機で実行する処理と変わっていないからである．そこで，マスタプロセスによって生成されたジョブを，使用するすべての計算機で分配し部分頻出パターンの抽出を行うために，マスタプロセスとする計算機には同時にスレーブプロセスも動作させる．つまり，計算機数が 2 の場合には 1 台の計算機でマスタプロセスとスレーブプロセスの両プロセスを動作させ，別のもう 1 台にはスレーブプロセスを動作させる．2 台の計算機で実際には 3 個のプロセスを動作させる．同様に計算機数が 4 の場合では 5 個のプロセスを動作させ，計算機数が 8 の場合では 9 個のプロセスを動作させている．これにより，部分頻出パターンの抽出をすべての計算機で行うことができる．

本評価実験では，(1) Kringle において閾値を 2 ，支持率を 40 ，ワイルドカード数を 7 とした場合に，(2) Zinc Finger において閾値を 2 ，支持率を 40 ，ワイルドカード数を 5 とした場合に，計算機の台数を 2 から 8 に増やしていくことで，どの程度の性能向上比が得られているかを図 4 に示す．また，計算機の台数

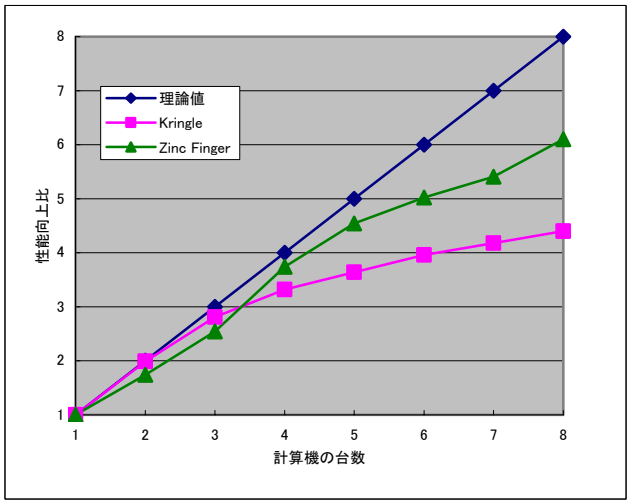

図 4　性能向上比

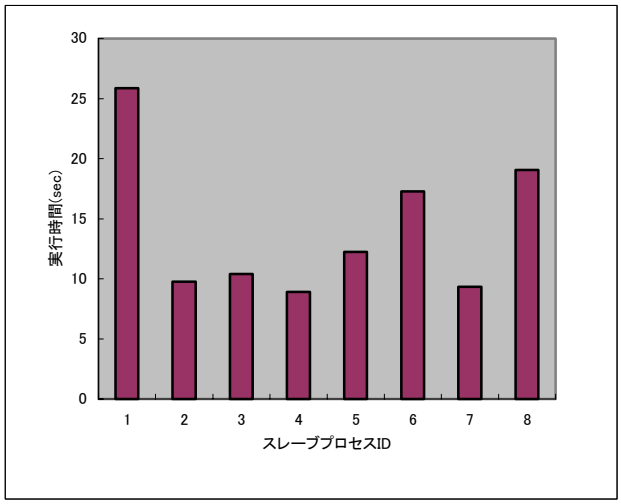

図 5　計算機数 8 のときの各スレーブプロセスの実行時間(Kringle)

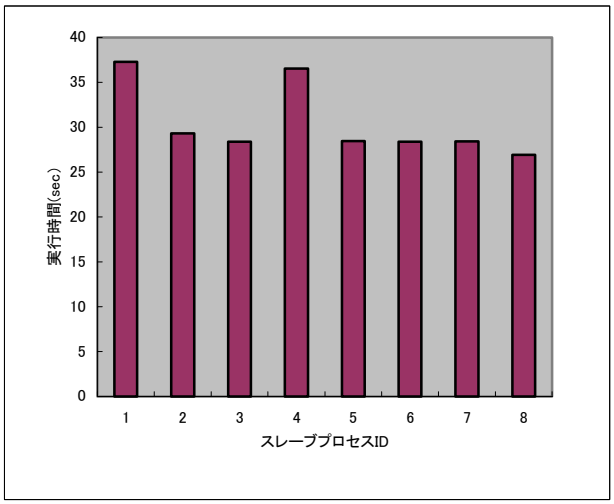

図 6　計算機数 8 のときの各スレーブプロセスの実行時間(Zinc Finger)

が 8 のときの各スレーブプロセスの実行時間を図 5 と図 6 に示す.

これらの結果より最大で，計算機数 2 のときには約 2 倍，計算機数 4 のときは約 3.7 倍， 8 のときは約 6 倍の性能向上比が得られた.効率のよい並列処理とは，N 台の計算機で並列にした場合，実行時間を 1／N に短縮することである．つまり理論的には計算機の台数を 8 台で実行する場合,実行時間は 1／8 になる．本実験では実行時間は最大で 1／6 になっている.この差として通信のオーバヘッドが考えられるが，本実験で使用したデータは高々数十キロバイトなので通信のオーバヘッドでこれだけの差がでるとは考えられない.

この理論値と離れている理由を図 5 と図 6 で説明する．図 5 と図 6 では，各スレーブプロセスの実行時間に差が生じていることがわかる．このことより負荷が分散さていないことがわかる．特に，図 5 ではもっとも早く処理を終えたプロセスと，もっとも遅く処理を終えたプロセスの実行時間で,約 3 倍の差が生じていることがわかる.閾値を下げて実験した場合には，これよりさらに差が生じたこともあり，極端に遅いスレーブプロセスが出てくる場合があることがわかった.

これは，送られてきたジョブを部分的に抽出する頻出パターンの数に差があるためだと考えられる．例えば，400 個のジョブを生成したとして，そのうち 399 個のジョブはほとんど抽出が続かず,残り 1 個のジョブは次々に抽出が続く場合である．この場合，この 1 個のジョブをさらに細かく分ければよい.

5.2. 評価実験－2

本評価実験では,(1) Kringle において閾値を 2 とし，支持率を 40 で固定しワイルドカード数を変化させた場合と,ワイルドカード数を 7 で固定し支持率を変化させた場合に,(2) Zinc Finger において閾値を 2 とし，支持率を 50 で固定しワイルドカード数を変化させた場合と,ワイルドカード数を 5 で固定し支持率を変化させた場合に,それぞれ計算機の数を 2 台,4 台,8 台と増やして実験した結果を図 7 から図 10 に示す.

図 7 から図 10 の結果によると，ほぼすべての結果において，支持率やワイルドカード数にかかわらず，計算機の数が増えるごとに計算時間の短縮を図れていることがわかる．これらの結果より，計算機の数を増やすことで支持率やワイルドカード数にかかわらず台数効果が得られているといえる．特に計算機の台数 1 の場合の Modified PrefixSpan 法による頻出パターンの抽出処理時間がながければながいほど，台数効果が現れていることがわかる．しかし，Modified PrefixSpan

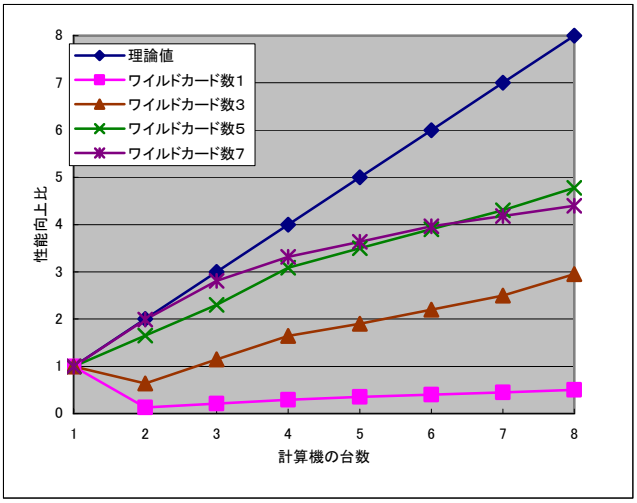

図 7　支持率を固定させた実験結果(Kringle)

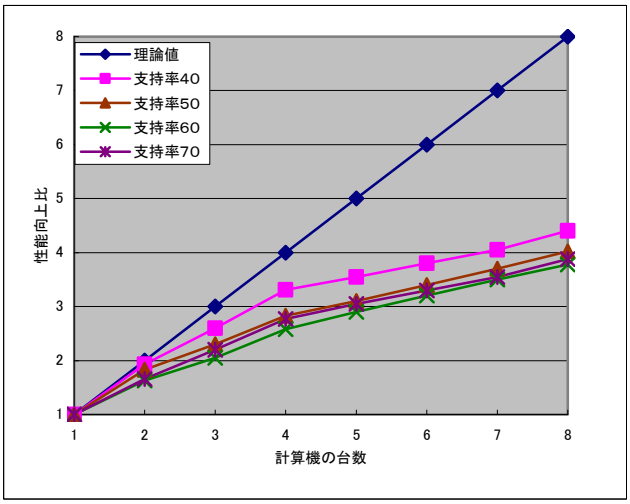

図 8　ワイルドカード数を固定させた実験結果(Kringle)

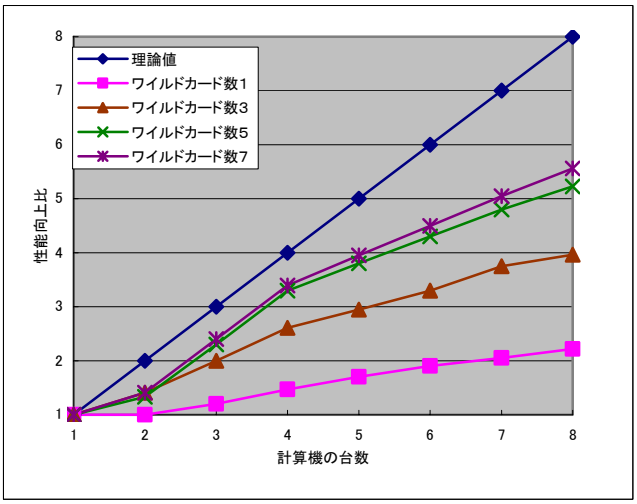

図 9　支持率を固定させた実験結果(Zinc Finger)

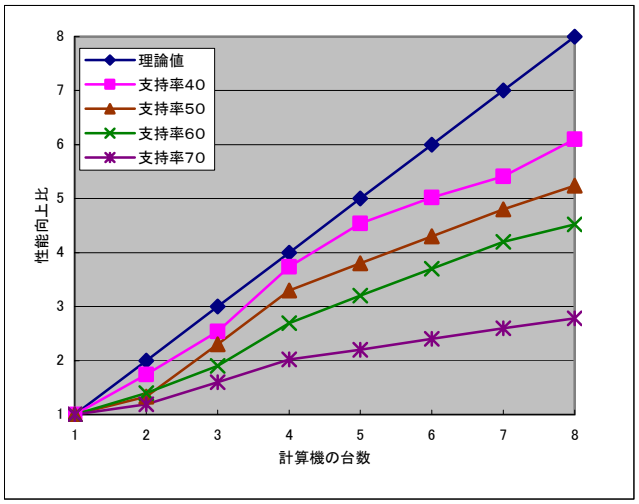

図 10　ワイルドカード数を固定させた実験結果 (Zinc Finger)

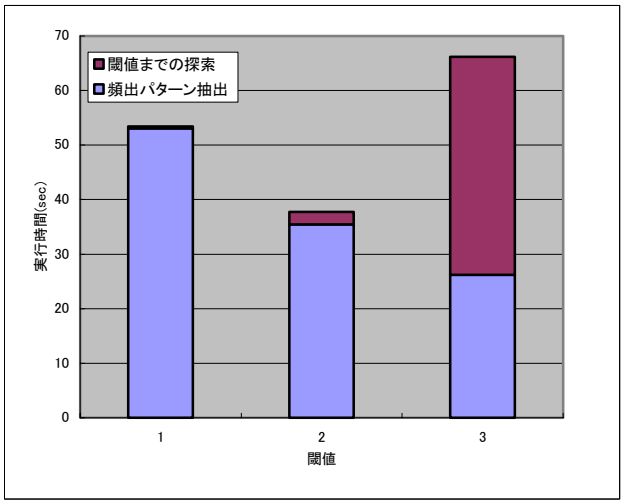

図 11　閾値を変更した場合の閾値までの探索時間と頻出パターンの抽出時間

法での頻出パターンの抽出処理が短い場合は，台数効果が得られていない場合もある．この理由は，部分頻出パターンの抽出より通信時間がかかることや，マスタプロセスが閾値までの探索の時点でほとんどの頻出パターンを抽出し終えた場合などがあげられる．

5.3. 評価実験－3

前項までの実験で明らかになったのは，ジョブに続く頻出パターンの数に差があることで，スレーブプロセスの実行時間に差があり，効率よく負荷分散が行えていないことであった．そこで，閾値を上げることでジョブをより多く生成し，負荷の分散を図る実験を行った．本評価実験では， Zinc Finger において支持率を 40，ワイルドカード数を 5，計算機の台数を 8 で固定し，閾値を 1，2，3 と変化させた場合，閾値までの探索時間と残りの頻出パターン抽出の実行時間を計測した結果を図 11 に示す．また各閾値の場合におけるスレーブプロセスの実行時間を図 12 に示す．

図 11 より，閾値を増やすことで，スレーブプロセスが行っている部分頻出パターン抽出の実行時間が減っているが，それ以上にマスタプロセスで行っている閾値までの探索時間が増えていることがわかる．図 12 によると，閾値が 1 の場合は各スレーブプロセスでの処理時間に大きな差があるが，閾値が 3 の場合はほとんどなくなっている．閾値の値を上げることにより，ジョブの均一化を図ることができるが，閾値の値は少しの変更で計算時間が莫大に増えてしまい，ジョブを多く生成するだけでは効率のよい負荷分散が行えないことがわかる．

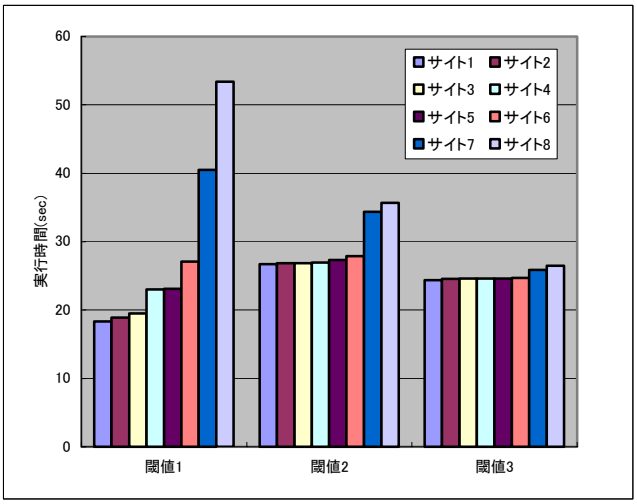

図 12 各スレーブプロセスの実行時間

以上の結果により，閾値を増やしジョブを多く生成するだけでは効率のよい負荷分散が行えないことがわかる．

6. 動的負荷分散方式の検討

4 章の検証実験により計算機数が増えることにより，台数効果が得られていることが示された．同時に，各スレーブプロセスの実行時間が均一でなく，なかには極端に遅いプロセスが出るという課題が生まれた．Modified PrefixSpan 法は，支持率やギャップ数を任意に変更できるために，これらの値によっては深さだけでなく幅も大きく変わってくる．このために，Modified PrefixSpan は図 2 のように，複数のジョブを作るだけでは，各計算機が行う部分頻出パターン抽出に差ができてしまい，負荷を均一にすることは難しいことがある．

この課題を解決するための Modified PrefixSpan 法の並列方式について検討を行った．全体の抽出を部分頻出パターン抽出 A に分割して処理を小さく分け，各計算機に配るだけでなく，ジョブを任された各計算機においてさらに部分頻出パターン抽出 B_i に分ける (i はサイトの id). 各計算機は B_i の探索を終えると，A から次のジョブを取り出す. A の仕事がなくなると，まだ処理を続けている計算機が持っている B_i を横取りして，各計算機に配る.

本手法では，単に部分木の処理の大きさだけでなく，例えば，各計算機の計算機能力が異なっていたとしても，負荷の均一化を図ることができる.

現在検討しているアルゴリズムの実装を図 6 に示す.

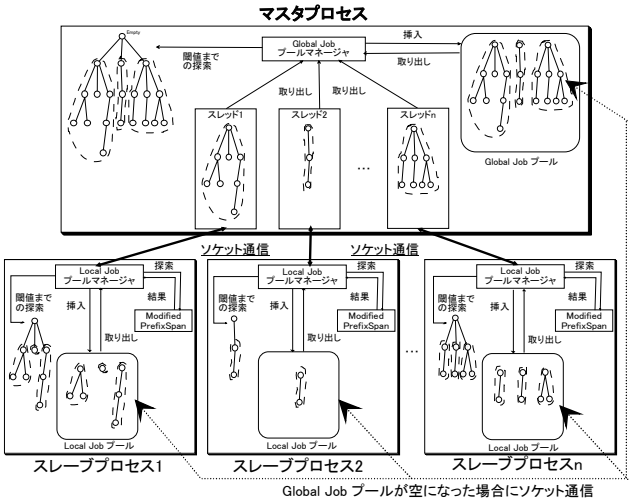

図 13　動的負荷分散方式のシステム構成

前述の 3. で示した詳細に加え，スレーブプロセスにも Job プールを作る，以下 Local Job プールとする. ジョブが送られてきたスレーブプロセスは，マスタプロセスと同様に閾値までの部分頻出パターン抽出を行い，得られるジョブを Local Job プールに挿入し，Modified PrefixSpan 法によりモチーフの抽出を続ける. マスタプロセスのジョブをすべてスレーブプロセスに送信し，Global Job プールが空になったとき，処理すべきジョブがまだ残っている Local Job プールからジョブを取り出し，Global Job プールに挿入する．２つの Job プールを連携させることでさらにきめ細やかな負荷分散が行える.

ジョブの大きさがわかれば計算機の性能によってジョブを配分する方式[7]で高い性能向上を得たという報告もある．Modified PrefixSpan 法では残念ながら，ジョブを生成した時点では抽出に正確にどのくらいの時間がかかるかわからない. 結果として，この 1 個のジョブを受け取ったスレーブプロセスは，他のスレーブプロセスに比べ計算時間が極端に遅くなってしまう.

7. おわりに

本論文では，並列 Modified PrefixSpan 法を実現するために，頻出パターン抽出を複数の小さなジョブにわけ，部分的に頻出パターンを抽出する方法について説明した. 実際の PC クラスタ上に実装をおこない検証実験をおこない台数効果があることを確認した．検証実験で用いたアミノ酸配列は小規模なものであり，さまざまなアミノ酸配列で検証していく必要がある. また実験で明らかになったのは，極端に重いジョブがある場合や，ユーザの経験的な閾値の設定では負荷を均一にできないことであった. 今後は 5 で提案した動的負荷分散処理や，自動的に最適な閾値を設定する方式を用いて，効率のよい並列 Modified PrefixSpan 法を完成させ，性能評価をおこなっていく予定である.

なお，本研究の一部は日本学術振興会・科学研究費(基盤研究(c)，課題番号 : 12680394) および広島市立大学・特定研究費(科研費奨励研究費(コード番号 : 9856)，科研費助成研究費(コード番号 : 0044)) の支援により行われた.

文　献

[1] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, Proc. of International Conference on Data Engineering (ICDE 2001), IEEE Computer Society Press, p215-p224, 2001.

[2] Tomoki Kanbara, Yasuma Mori, Hajime Kitakami, Susumu Kuroki and Yukiko Yamazaki: Discovering Motifs in Amino Acid Sequences using a Modified PrefixSpan Method, Currents in Computational Molecular Biology 2002, ACM-SIGACT, Washington DC, pp.96-97, April 2002.

[3] 蒲原朋樹，森康真，北上始，黒木進 : PrefixSpan 法を用いたモチーフ発見システム，第１３回データ工学ワークショップ (DEWS2002)，電子情報通信学会データ工学研究専門委員会，Online Proceedings(http://www.ieice.org/iss/de/DEWS/proc/2002/proceedings.html)，2002 年 3 月.

[4] Timothy L. Bailey and Charles Elkan, Fitting a mixture model by expectation maximization to discover motifs in biopolymers, Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, pp. 28-36, AAAI Press, Menlo Park, California, 1994.

[5] 金久 實:ポストゲノム情報への招待，共立出版社，2001 年

[6] Inge Jonassen, John F. Collins, and Desmond G. Higgins: Finding flexible patterns in unaligned protein sequences, Protein Science, pp.1587-1595, Cambridge University Press, 1995.

[7] 荒木拓也，村井均，蒲池恒彦，妹尾義樹 : データ並列言語を対象とした動的負荷分散機構の実現と評価，情報処理学会論文誌 : Vol. 43 No. SIG 6(HPS5) ハイパフォーマンスコンピューティングシステム，pp.66-75，Sep. 2002