ハイパフォーマンス
コンピューティング 87－10
（２００１．７．２６）

多目標追尾アルゴリズム航跡型 MHT の並列化
－解候補生成の並列化とその評価－

佐藤裕幸[I]，中島克人[II]
I 三菱電機（株）情報技術総合研究所，II 三菱電機（株）本社開発業務部

航跡型 MHT は、探知データと目標の相関を一定個数の複数の仮説として保持しながら追尾処理を行うので、高い追尾性能を示すという特長がある。我々は、仮説生成処理を高速化することで保持可能な仮説数を増加させ、更に追尾精度を向上させるために、仮説生成部分を分散メモリ型並列マシン上で並列化した。並列化に際しては、負荷の均等化、通信オーバヘッド軽減、プロセッサの遊休状態回避等を考慮している。その結果、キャッシュメモリの増量効果もあり、6 プロセッサを使用して、全実行時間で 4 倍～11 倍、並列化した仮説生成部分に関しては 8 倍～11 倍のプロセッサ台数を超える高速化を確認した。

Parallelizing of Multiple Target Tracking Algorithm Track-Oriented MHT
- Parallelizing of a Hypothesis Creation Process and its Evaluation -

Hiroyuki Sato[I], Katsuto Nakajima[II],
I Mitsubishi Electric Corp. Information Technology R&D Center,
II Mitsubishi Electric Corp. Planning & Administration Dept.

In order to raise tracking accuracy, we decreased the time for track-oriented MHT, multiple hypothesis tracking, which is a multiple target tracking algorithm, by using a network parallel processing. In parallel processing, we considered a load balancing, decrease of communication overhead and preventing processors from being idle status. The total execution time produced a 4 to 11 times increase in speed with 6 CPUs over the sequential execution and the time for parallelized hypothesis creation process produced a 8 to 11 times increase, because of increase in quantity of a cache memory.

1 はじめに

目標追尾とは、レーダ等のセンサの観測結果から目標の航跡を抽出し、位置，速度等の運動諸元を推定する問題である。目標追尾の抱える課題の１つに、追尾目標の近傍に他の目標やクラッタ等の不要信号が存在する高密度環境下での追尾性能の確保が挙げられる。高密度環境下では、センサから得られた観測値のいずれが各追尾目標のものであるかを判定する相関処理が重要となる。従来は、各追尾目標に対して、予測位置に最も近い観測値を選択し、追尾計算に使用する方式が採られていたが、この方式では高密度環境下では追尾性能が不十分であった。そこで、より高度な相関方式を採用した航跡型 MHT（Multiple Hypothesis Tracking）[1] が提案された。

この航跡型 MHT では、観測値の時系列データから成る航跡のいずれが追尾目標であるかについて、他の目標やクラッタが存在する可能性及び追尾目標が探知されなかった可能性も考慮して、複数の仮説を構成しながら追尾処理を行う。このように複数の仮説を保持しながら処理を進めていくため、追尾精度は優れるが、処理負荷が高く、また仮説の数が組合せ爆発を起こし膨大になる危険性があるという問題点があった。そこで、仮説数を抑制するために、信頼度の高いものから順に固定数の仮説しか生成しない N ベスト探索アルゴリズムを用いることで、航跡型 MHT が高速化されている[2] [3] 。

この N ベスト版航跡型 MHT において、追尾精度を向上させるには、N の値、すなわち仮説生成上限数をできるだけ大きくしなければならない。そのためには、生成仮説数を抑えた N ベスト版航跡型 MHT においても、処理の高速化は重要な課題である。そこで我々は、並列処理により、N ベスト版航跡型 MHT の高速化を行っている。本稿では、この並列化方式及びその評価について報告する。

2 航跡型 MHT の概要

図 1の処理フローに基づいて、航跡型 MHT の概要を

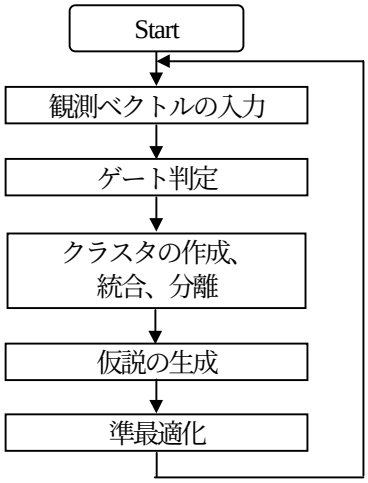

図 1 航跡型 MHT の処理フロー

述べる。

2.1 観測ベクトルの入力

1サンプリング時刻の観測ベクトル（センサによる目標の X,Y,Z 座標など）を入力する。

2.2 ゲート判定

各サンプリング時刻から高々一つの観測ベクトルを選んだ時系列データ（0 観測ベクトルは、目標が探知できない場合を想定）を航跡と呼ぶ。1サンプリング前のどの航跡とどの観測ベクトルが対応しているのかの相関処理を行う範囲をゲートと呼ぶ。すなわち、各1サンプリング前の航跡に対して、現サンプリング時刻の観測ベクトルが存在すると予測される空間の領域をゲートと定め、この領域の観測ベクトルのみ相関処理の対象とする。

2.3 クラスタの作成、統合、分離

互いに観測ベクトルを共有しない航跡の集合をクラスタと呼ぶ。すなわち、航跡 T_i と T_j が少なくとも一つの観測ベクトルを共有する場合に限り、航跡 T_i と T_j を類似航跡と呼び、$T_i \sim T_j$ と書く。航跡 T_i, T_j において

$$T_i = T_{i1} \sim T_{i2} \sim \cdots \sim T_{in-1}, T_{in} = T_j \quad (1)$$

となる $T_{i1}, T_{i2}, \cdots, T_{in}$ が存在する場合に限り

$$T_i \equiv T_j \quad (2)$$

と定義する。各サンプリング時刻の全航跡は、式(2)の関係により、互いに素な複数の部分集合に分けられ、この部分集合をクラスタと呼ぶ。以下の2.4, 2.5の処理は、各クラスタ毎に独立して処理することが可能である。

どのクラスタにも属さない観測ベクトル（どの航跡のゲート内にも入らない観測ベクトル）がある場合は、新たにクラスタを生成する。また、異なるクラスタに属する複数の航跡のゲート内に同じ観測ベクトルが入っている場合、それらのクラスタの統合を行う。更に、後述の準最適化処理により過去のデータを捨てた結果、互いの航跡が共有する観測ベクトルを持たなくなった場合は、クラスタを分離する。

2.4 仮説の生成

各クラスタにおいて、0 個以上の航跡を選択して構成した航跡の集合を仮説と呼ぶ。各仮説は、クラスタ内のどの航跡の組合せを選ぶのが正しいのかを表わすものであり、最終的に求めたい目標の位置、速度等の運動諸元の基となる解候補である。各仮説を構成する際の基準は、同一仮説内に属する複数の航跡間で観測ベクトルを共有しないように航跡を選択することである。なお、仮説に含まれるどの航跡にも属さないクラスタ内の観測ベクトルは、この仮説において誤信号と扱ったことに相当する。

仮説を生成するには、まず現サンプリング時刻における観測ベクトルの入力を反映した航跡を生成して、それらを選択する。これは、各観測ベクトルが、どの既存の航跡と対応するのか、新たな航跡なのか、誤信号なのかを表わす選択木として考えることができる。

以下に、例を用いて説明する。1サンプリング前のある仮説の航跡が T_1 及び T_2 であり、現サンプリング時刻では、航跡 T_1 がゲート内に2つの観測ベクトル Z_1 及び Z_2 を捕らえ、航跡 T_2 が1つの観測ベクトル Z_2 を捕らえている（図 2）。この状況下での航跡と観測ベクトルの可能な組合せは、図 3に示すような選択木で表わすことができる。この木の根から各葉までを辿る枝がそれぞれの仮説であり、この例では 11 個の仮説

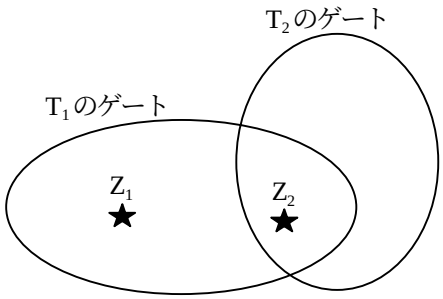

図 2 ゲートと観測ベクトル

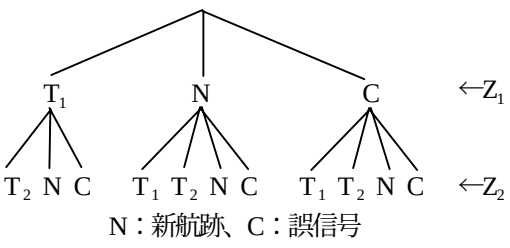

図 3 航跡と観測ベクトルの組合せ

が生成されることになる。

１つの親仮説から生成される子仮説の数は、その仮説内の航跡と観測ベクトルとの可能な組合せ数になるので、航跡数や観測ベクトル数が少しでも多くなると、直ぐに組合せ爆発を起こしてしまう可能性がある。そこで、仮説の生成段階で信頼度の高いものから所定数しか生成しないようにしている。

前述のように、考えられる仮説は図 3のように選択木で表わすこともできるが、各列を誤信号、既存航跡、新航跡、各行を観測ベクトルとし、各要素を信頼度（相関度）とする行列とし、互いに同一の列からの要素を選択しないように各行について１要素ずつ選択することでも、仮説を表わすことができる。そして、選択した各要素の値の総和が最も大きくなるように選択する方法を見つけることが、最も信頼度の高い仮説を生成することになる。このような選択問題を効率的に行うアルゴリズムとして Munkres のアルゴリズム[5]がある。更に、最高信頼度仮説を生成した後は、選択した枝を境界とする部分木に分割する。例えば図 4において、太い枝が選択された仮説であるとすると、その枝を境界とする部分木は、四角で囲った 2 個所となる。そして、それぞれの部分木で最高信頼度仮説を生成した後、それらの中で最も信頼度が高い仮説を選択する。これを仮説数が規定数 N になるまで繰り返す。この方法を Murty アルゴリズム[6] と呼び、これにより効率的に高信頼度の仮説から生成することができる。

なお、仮説の信頼度、すなわちその仮説が真である確率は、文献[4] に詳しいが、目標の探知確率、ゲート内捕捉確率、誤信号発生率、新目標発生率、仮説を構成する航跡と対応する観測ベクトルとの相関度及び基となる親仮説の信頼度の項から構成されている。

2.5 準最適化

古いデータを捨てることにより、最新 N サンプリングの観測ベクトル構成が同一である航跡を同一視した結果、内容（構成する航跡）が同一となった仮説同士を統合する。これを N スキャンリミットと呼び、仮説数の抑制を信頼度ではない別の観点で行っていることに相当する。

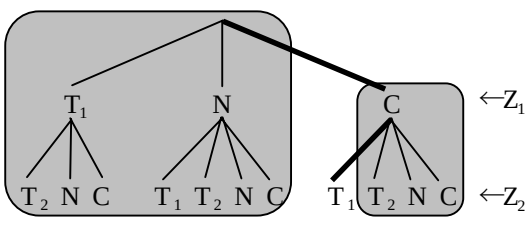

図 4 選択木の分割

3 航跡型 MHT の並列化

航跡型 MHT では、個々のデータ（観測ベクトル、クラスタ、航跡、仮説など）毎に独立して処理しているのではない。従って、全体を統一的に並列化することはできず、それぞれの処理を個別に並列化するしかない。そこで、どの処理に時間を要しているのかを解析し、最も負荷の高い処理から並列化することにした。

3.1 負荷解析

図 5に各サンプリング時刻毎の実行時間（左軸で棒グラフ）と仮説生成の全実行時間に対する割合（右軸で折線グラフ）を示す（生成仮説上限数は 3,000 個）。棒グラフの黒い部分が仮説生成の時間であり、圧倒的に仮説生成に時間を要していることが分かる。実行時間を要しているサンプリング時刻 24 秒以降の仮説生成時間の割合の平均は、92%である。なお、「その他」の内で最も負荷の高い処理は、クラスタ統合である。クラスタ統合では、お互いのクラスタ内の仮説同士を組合せるので、組合せ数が多くなると処理時間がかかっている。

以上の負荷の解析結果から、仮説の生成部分を並列化することにした。逐次の航跡型 MHT プログラムにおけるこの仮説の生成方式は、以下の通りである。

(a) 全ての親仮説について、それぞれの最も信頼度の高い子仮説を生成する。
(b) N 個の親仮説から生成された N 個の子仮説の中で最も信頼度の高い子仮説を採用する。採用された子仮説を生成した親仮説は、次に信頼度の高い子仮説を生成する。
(c) (b)の処理を子仮説が N 個になるまで繰り返す。

3.2 並列化方式

この仮説生成において、各親仮説で子仮説を生成する過程は独立して実行できる。そこで、その親仮説毎の子仮説生成の部分を以下にように並列化する（図 6）。なお、子仮説を生成するプロセッサをクライアントと呼び、各クライアントで生成された子仮説を信頼度の

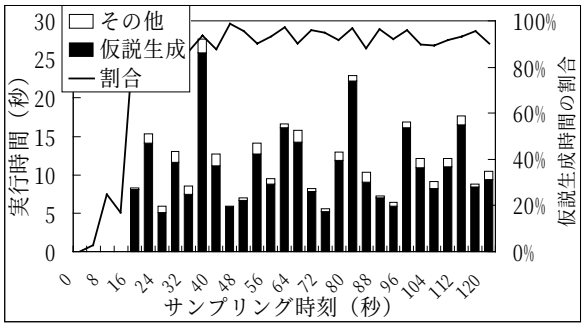

図 5 各サンプリング時刻毎の実行時間

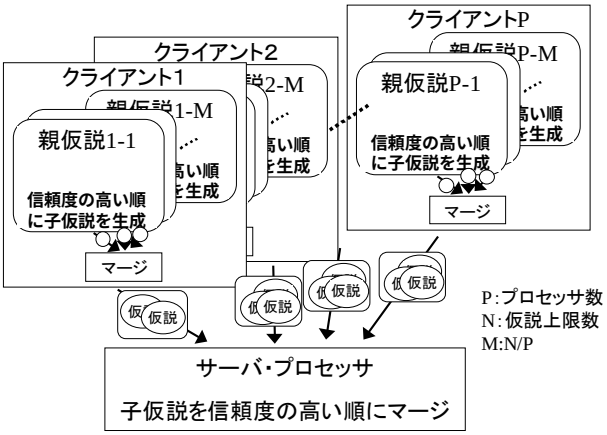

図 6 仮説生成の並列化

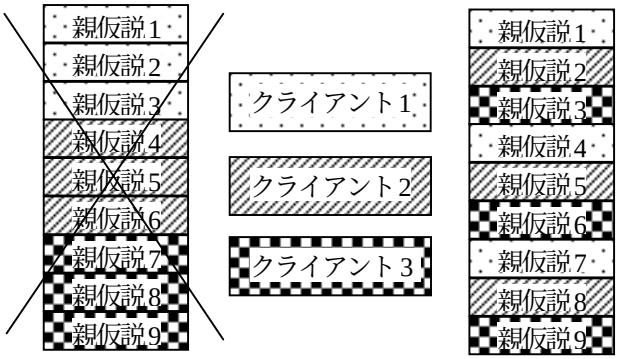

図 7 親仮説の分配方法

高い順にマージするプロセッサをサーバと呼ぶことにする。サーバの処理は子仮説のマージだけであり負荷が低いので、サーバのプロセッサはどれか１つのクライアントのプロセッサと同じあっても良い（ある１つのプロセッサはサーバでありクライアントでもある）。

(a) N 個の親仮説をクライアント台数 P で割り、各クライアントに分配する。

(b) 各クライアントでは、サーバからの要求に基づき、分配された親仮説群を基に逐次版と全く同様に子仮説を信頼度の高い順に幾つか生成し、それらをサーバへ送信する。

(c) サーバは、各クライアントから送られてきた子仮説群（各クライアント内では信頼度順にソートされている）を信頼度順にマージ・ソートする。このマージの際に、あるクライアントからの子仮説群を使い果たしたら、次の子仮説の生成をそのクライアントに要求する。

(d) (c)のサーバでのマージ処理を子仮説数が N 個になるまで繰り返す。

この並列処理方式では、親仮説から信頼度の高い順にある個数の子仮説を生成するという単位で並列化しているので、子仮説の生成方法（Munkres アルゴリズムや Murty アルゴリズム）には全く依存していない。従って、その生成方法を別のアルゴリズムに変更したとしても、本並列処理方式はそのまま適用できる。

3.3 並列化方式の特長

(1) ラウンドロビンによる静的負荷分散

親仮説の各クライアントへの分配は、静的に割り付ける（処理の最初に割り付け、負荷状況によって移動させない）。前述したように、子仮説の信頼度は、親仮説の信頼度と航跡と観測ベクトルの相関度等から算出される。そのため、親仮説の信頼度がそこから生成される子仮説の信頼度に影響する。従って、信頼度の高い順に並んでいる親仮説を N をクライアント数 P で割ったブロック毎に各クライアントに割り付けてしまうと、上位のクライアントに高信頼度の親仮説が集中し、それらのクライアントから生成される子仮説のみが採用される可能性が高くなり、クライアント間で負荷のアンバランスが起き易くなる。

そこで、ブロックではなく、高信頼度の親仮説から１つずつ順に各クライアントへ割り付けるようにする（図 7）。これにより、各クライアントの親仮説の信頼度が均衡化され、それに従って各クライアントの負荷も均等になる。

(2) 複数仮説単位での通信

各クライアントは子仮説を一つ生成したら直ぐにサーバへ送るのではなく、複数個生成してから送信する。これは、子仮説を一つ生成する処理が粒度として小さすぎる可能性があり、一度に生成する子仮説の数でその粒度を調整できるようにするためである。現在、一度に生成する子仮説の数は、N/P/10 としている（N：親仮説数、P：クライアント数）。すなわち、各クライアント毎に平均 10 回程度サーバへ子仮説群を送信することになる。この数は、本来、並列処理環境のプロセッサ速度と通信速度の比率により調整すべきものである。

(3) 仮説生成のダブル・バッファリング

各クライアントは、生成した子仮説群をサーバへ送信した後は、次の生成要求がサーバから来るまで暇となり、クライアント・プロセッサが遊休状態になる可能性がある。そこで、各クライアントは、次にサーバから子仮説生成の要求が来る前に、次の子仮説群を前もって生成して溜めておく（バッファリングする）ようにしている。これにより、各クライアントの遊休状態が緩和されるだけでなく、早めに生成を開始しているので、サーバからの生成要求に対してのレスポンスも早くなる。

4　評価実験

今回の並列化による効果を評価するために、シミュレーションによる計測実験を行った。誤信号密度及び新目標密度は同じ値を取り、それぞれ低、中、高の 3 種類のデータを用いた。サンプリング間隔は 4 秒で、シミュレーション時間は初探知の時刻 0 秒から 120 秒までであり、測定は乱数のシードを変えて 30 回行い、その平均を取った。また、仮説の生成上限数（=N）は 3,000 個である。実験に用いた並列処理環境を表 1に示す。Alpha21264 プロセッサを 2 つ持つマシンを Myrinet で 3 台接続した環境である（CPU 総数は 6）。本計測でクライアント台数を増やす場合は、各マシンに 1 つずつ割り当てていき、次に 2 つ目の CPU に割り当てるようにしている。

4.1　実行時間

表 2に、各クライアント数毎の 1 サンプリング時刻当たりの全実行時間及び仮説生成時間の平均を示す。クライアント数が 0 とは、従来の逐次の航跡型 MHT プログラムでの実行時間であり、クライアント数が 1 とは、仮説の生成のみを別プロセッサで実行した場合の実行時間である。また、図 8に、各クライアント数毎の高速化率（全実行時間の逐次プログラムとの比率）及び並列化効果（仮説生成時間の 1 クライアントとの比率）を示す。これらの結果から、以下のことが言える。

- クライアント数が 1 台の場合は、仮説の生成が別プロセッサで実行されるだけで、並列実行は全く行われないし、仮説生成に必要な入力データと生成された仮説データを通信しているにもかかわらず、逐次プログラム（クライアント台数 0 台）より速くなっている。同様に、高速化率及び並列化効果ともに、クライアント数以上の効果が出ている場合がある。
- サーバと一つのクライアントが同一 CPU で動作するクライアント数 6 の場合は、返って遅くなっている。別途 MPI を使用しない版で計測した場合は、遅くなっていなかったので、MPI の共有メモリ機能が関係しているのではないかと考えられる。
- 誤信号及び新目標密度が高くなると組合せ数が多くなるので、全体及び仮説生成ともに実行時間は長くなるが、並列化効果は密度によってそれほど変わらない。高速化率は、密度が低い方が効果が出ている。これは、密度が高くなると仮説生成以外の処理割合が増えているからだと考えられる。
- 並列化効果は、ほぼ直線になっており、更にクライアント（プロセッサ）数を増やしても効果は持続すると考えられる。ただし、高速化率の方は密度が中以上で頭打ちの傾向が見られる。これは、今回並列化しなかった部分（恐らくクラスタ統合）が次にボトルネックになっているからだと思われる。

前記のクライアント数以上の効果が出ている理由としては、プロセッサ数を増やすことによりキャッシュメモリ量も増加し、キャッシュヒット率が向上するためであると考えられる。すなわち、クライアント数が増えるに従って個々のクライアントで扱う仮説

表 1 並列処理環境の主な仕様

CPU/クロック	(Alpha21264/667MHz×2) ×3
メモリ	512MB
キャッシュ	1次:8KB, 2次:96KB, 3次(外部):2MB
OS	SuSE Linux 6.3
ネットワーク	Myrinet(MPICH-1.2.0, GM-1.1.3.14)

表 2 全実行時間と仮説生成時間

全実行時間（単位：秒）

クライアント数	0	1	2	3	4	5	6
誤信号密度:低	10.39	6.57	2.31	1.50	1.15	0.95	3.31
誤信号密度:中	12.52	10.49	3.89	2.58	2.20	1.78	4.46
誤信号密度:高	17.05	16.75	7.67	5.27	4.80	4.16	7.34

仮説生成時間（単位：秒）

クライアント数	0	1	2	3	4	5	6
誤信号密度:低	9.98	6.20	1.94	1.13	0.78	0.58	2.76
誤信号密度:中	11.61	9.53	2.92	1.62	1.23	0.81	3.00
誤信号密度:高	15.21	14.69	5.60	3.19	2.73	2.07	4.21

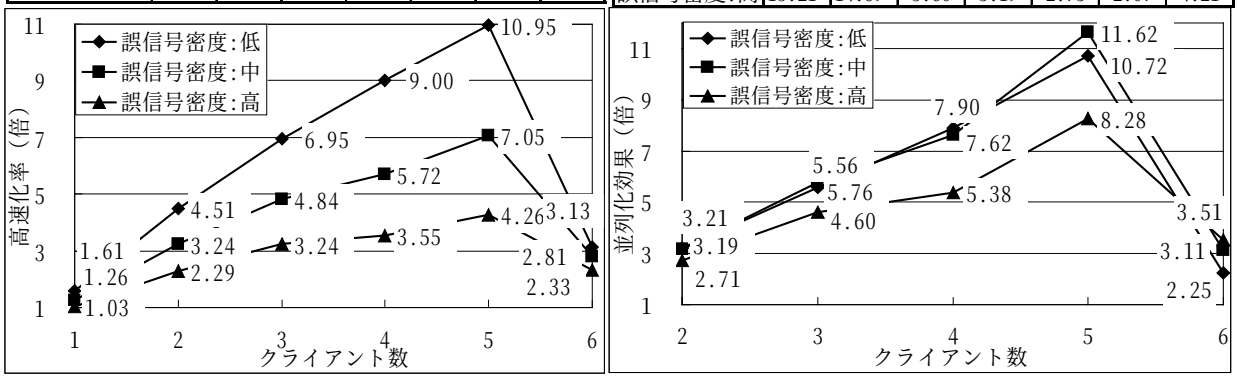

図 8 全実行時間の高速化率と仮説生成部分の並列化効果

数が少なくなるが、クライアント当たりのキャッシュメモリ量は変わらないので、キャッシュヒット率が向上するのである。

4.2 通信量と負荷の均等具合

次に、並列化効果を妨げる要因である通信オーバヘッドと各プロセッサでの負荷の不均衡を調べるために、サーバと各クライアント間での通信量と各クライアントで生成した子仮説の数を計測した。なお、計測に用いたデータは、前節の誤信号密度:中であり、クライアント数は5台である。

(1) 入力データ（サーバ→各クライアント）量

仮説生成を始める前にサーバから各クライアントに送られる仮説生成に必要なデータは、どの仮説がどの航跡を保持しているかという情報と各航跡と観測ベクトルの相関度情報である。前者はバイトデータ（本来はビットデータでも可）の2次元配列であり、後者はダブルワードの2次元配列である。それぞれの要素数は、仮説数、航跡数、観測ベクトル数で決まる。

この入力データの1クライアントの1並列実行当たりの平均データ量は、89K バイトであった。通信には Myrinet の MPI を使用しており、実効性能は別途計測により 134MB/s（レイテンシーは 40μ秒）なので、この入力データの通信時間は 0.7 ミリ秒程度である。5台実行で仮説生成時間が 810 ミリ秒なので、この通信時間を5倍したとしても、通信は全くオーバヘッドにはなっていないと言える。

(2) 出力データ（各クライアント→サーバ）量

出力データは生成した子仮説の情報そのものであり、基となった親仮説の識別子と航跡と観測ベクトルの組である。

この出力データの1クライアントの1並列実行当たりの平均データ量は、16K バイトであった。これは、入力データよりかなり小さいので、全くオーバヘッドはないと考えて良い。

(3) 子仮説生成数

各クライアントで生成する子仮説の数を計測し、それが各クライアントでどの程度異っているかを調べた。その結果、各クライアントで生成する子仮説の数は、1 クライアントの 1 並列実行当たりの平均で 657 個であった。これは、仮説上限数 3,000 個をクライアント数 5 で割った 600 に、先行して生成した数 60 を加えた数 660 に近い数字である。また、各クライアントの生成数の標準偏差（ばらつき）の平均は、55 個であった。これは、657 個に対して 12%であり、静的負荷分散にもかかわらず、それほど負荷が不均衡になっていないと言える。

次に、ラウンドロビンによる分配の効果を知るために、ブロック割付した場合も計測した。その結果、各クライアントで生成する子仮説の数の平均は変わらないものの、標準偏差は 963 個と圧倒的に大きくなっていた。また、全実行時間は 1.78→2.16 秒、仮説生成時間は 0.81→1.20 秒に増えており、ラウンドロビン分配による効果を確認することができた。

5 おわりに

以上、信頼度の高いものから順に固定数の仮説しか生成しない N ベスト版航跡型 MHT の並列化について報告した。並列化に際しては、最も負荷の高かった仮説生成部分を並列化し、各プロセッサで負荷が均等化されるように処理を分配し、並列処理粒度を調整することによりプロセッサ間の通信オーバヘッドが出ないようにし、各プロセッサでは先行して仮説生成を行うことにより遊休状態にならないようにしている。その結果、キャッシュメモリの増量効果もあり、6 プロセッサを使用して、全実行時間で 4 倍～11 倍、並列化した仮説生成部分に関しては8倍～11倍の高速化を確認した。

今後は、生成仮説数を増加させることによる追尾精度の評価を行っていく予定である。そして、計算能力と目標数や誤信号密度等の扱えるデータの関係を明らかにしていきたいと考えている。

参考文献

[1] 小菅義夫, 辻道信吾, 立花康夫, ”航跡型多重仮説相関方式を用いた多目標追尾,” 信学論(B-II), vol.J79-B-II, no.10, pp.677-685, Oct. 1996.

[2] 小幡康, 系正義, 辻道信吾, 小菅義夫, “N ベスト解探索アルゴリズムによる航跡型 MHT の高速化(1)－N ベスト解探索アルゴリズムの導入方式－,” 2001 信学総大, B-2-32, Mar. 2001.

[3] 系正義, 小幡康, 辻道信吾, 小菅義夫, “N ベスト解探索アルゴリズムによる航跡型 MHT の高速化(2)－演算時間および追尾維持性能の評価－,” 2001 信学総大, B-2-33, Mar. 2001.

[4] 系正義, 辻道信吾, 小菅義夫, “航跡型 MHT による追尾維持性能のシミュレーション評価,” 信学技報 SANE99-117, SAT99-154, P.71-78, Fep. 2000.

[5] F.Bourgeois, J.C Lassale:”An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices,” Comm. Of the ACM, vol.14, no.12, Dec. 1971.

[6] K.T.Murty, “An Algorithm for Ranking al the Assignments in Order of Increasing Cost,” Oper. Res., 16:682-687, 1968.