Non-Homogeneous 置換モデルを搭載した系統解析プログラムの MPI/OpenMP ハイブリッド並列化：大規模遺伝子データセットへの適用に向けて

石川奏太[†1, 2, a)]　中尾昌広[†3]　稲垣祐司[†2, 4]　橋本哲男[†2]　佐藤三久[†1, 3]

多様な生物種より得られた大規模遺伝子配列データをもとに分子系統解析を行う場合，パラメータの異なる複数の置換モデルを系統樹の各系統に適用する Non-Homogeneous 置換モデルが有効である．一方，この解析法では推定すべきパラメータ数と計算時間が飛躍的に増大するという問題が生じるため，系統解析プログラムの並列化が必須である．本研究では塩基配列データにおける系統間での G+C 含量の不均一性を許容する Non-Homogeneous 置換モデルを搭載した系統解析プログラム”NHML”を対象とし，系統樹の尤度計算アルゴリズムに MPI および OpenMP によるハイブリッド並列計算技術を導入した．シミュレーション配列を用いた性能評価では，1 本の系統樹の尤度計算において 256 並列まで良好な並列化効率が認められた．さらに MPI コミュニケータを分割することで，複数本の系統樹に対する尤度計算を並列的に行わせた．結果，1024 CPU コア以上を用いた場合であっても優れた並列性を実現した．

Hybrid MPI/OpenMP parallelization of a phylogenetic program with Non-Homogeneous models: toward the analyses of large-scale sequence datasets

SOHTA A. ISHIKAWA ,[†1, 2, a)]　MASAHIRO NAKAO ,[†3]　YUJI INAGAKI ,[†2, 4]
TETSUO HASHIMOTO ,[†2] and MITSUHISA SATO ,[†1, 3]

In the phylogentic analyses based on sequence datasets derived from diverse species, Non-Homogeneous models, which allocate different model parameters on each node of a tree, are efficient to reconstruct correct phylogenetic trees. However, the analyses with Non-Homogeneous models can be computationally intense because of an enormous amount of model parameters need to be optimized. Therefore, to accelerate phylogenetic analyses with Non-Homogeneous models, the parallelization of phylogenetic programs is necessary. In this study, we parallelized a phylogenetic program “NHML”, which implements a Non-Homogeneous model to take the heterogeneity of G+C content in nucleotide sequences among lineages in to account. We applied two approaches for parallel computing, OpenMP and MPI, into the algorithm for the calculation of the likelihood of a tree. From the analyses of simulated sequence datasets, this HYBRID version of NHML showed good parallel efficiency for the likelihood calculation of a tree until using 256 CPU cores. Moreover, we divided MPI communicator into several sub-communicators to conduct likelihood calculations of multiple trees in parallel. Consequently, we achieved the suitable performance of parallelization with more than 1024 CPU cores.

1. はじめに

分子系統解析では，現存する生物種から得られた遺伝子配列を材料とし，各配列が祖先配列からどのような順番で分岐し進化してきたかを表す進化系統樹を推測する．系統樹の推測法のうち最も広く用いられている最尤法では，遺伝子配列の進化はマルコフ過程を仮定し，祖先配列から各末端配列へと進化する確率(遷移確率)を計算し，配列データからある系統樹が実現する確率 = 尤度を算出する. 遷移確率は,「置換モデル」と呼ばれる，遺伝子配列における塩基・アミノ酸・コドン間の置換速度を表した速度行列を用いることで計算される．また，系統樹の尤度はその樹形(トポロジー)によって異なるため, 配列データより考えうる系統樹の樹形それぞれについて尤度を求め，それらを比較することで最大尤度をもつ系統樹(最尤系統樹)を最終的に選択する.

一般的な系統解析では，系統樹の尤度計算にはただ一つの置換モデルを用い,「遺伝子配列の進化プロセスは系統間で不変である」ことを仮定する．しかし，上の仮定は，いずれかの配列が他とは異なるプロセスに従って進化する可能性を許容できず，現実の配列データ中に「進化プロセスの不均一性」が存在する場合，データ中の実際の置換パターンと仮定した置換モデルとの間に著しい不整合が生じ，生物の真の系統を表す系統樹とは異なる系統樹(アーティファクト)が誤推測される[1,2]. 特に近年，シークエンシング技術の飛躍的な進歩により，広範囲にわたる生物種から

†1　筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of, Tsukuba, Tsukuba, Ibaraki 305-8577, Japan
†2　筑波大学大学院生命環境科学研究科
Graduate School of Life and Environmental Sciences, University of Tsukuba, Tsukuba, Ibaraki, 305-8577, Japan
†3　理化学研究所計算科学研究機構
RIKEN Advanced Institute for Computational Science
†4　筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba
a)　saishi@ccs.tsukuba.ac.jp

得られた遺伝子配列データに基づく大域的な分子系統解析が可能になったことで，進化プロセスの不均一性は頻繁に生じる問題となっている[3,4].

一方，パラメータの異なる複数の置換モデルを用意し，系統樹の各系統でモデルを変化させることで，遺伝子配列の進化プロセスが進化の各段階で変化することを許容する方法も考案されている[5,6]. このような「Non-Homogeneous 置換モデル(以下 NH モデル)」は，複雑な進化プロセスによって生成されたと考えられる配列データの解析には非常に有用であることが，シミュレーション解析ならびに実解析結果より示唆されている[7,8]．ただし，NH モデルでは用意するモデルの数や系統樹上でのモデルの変化点に比例して，推定すべきパラメータ数と系統解析に要する計算時間が飛躍的に増大するという問題が生じる．

大規模遺伝子配列データに基づく分子系統解析を高速化するために，現状では幾つかの系統解析用プログラムに対し，MPI/PTHREADS ハイブリッド並列化[9]や GPGPU 並列化[10]が行われているが，これらのプログラムには NH モデルが実装されておらず，専用プログラムの並列化が新たに必要である．本論文では実データ解析においても広く用いられる系統解析プログラム NHML[11,12]を対象とし，Message Passing Interface(MPI)および OpenMP によるハイブリッド並列化を行った．また，シミュレーションによって生成した塩基配列データを用いて，プログラムの速度向上についての評価を行った．

2. 最尤法による系統樹推測について

2.1 NH モデルによる系統樹の尤度計算

本節では，最尤法による系統樹推測について，NH モデルを用いた例をもとに説明する．簡単のため，図 1A に示したような taxon 1 ~ 4 の 4 種からなる塩基配列データ D より，図 1B に示される系統樹の尤度を計算する場合を考える．塩基配列データはアデニン(A)，チミン(T)，グアニン(G)，シトシン(C)の 4 つの塩基から構成され，N 種の生物について M 個の座位がアライメントされた $N \times M$ の行列として表現される．ここで，h 番目の座位において taxon 1 ~ 4 それぞれで塩基 p，q，r，s が観察されたとする．taxon 1，2 および taxon 3，4 の共通祖先における塩基をそれぞれ i，j，全ての配列の共通祖先を R とし，その塩基を x と仮定する．また，各塩基から次の塩基に遷移する時間は系統樹の各枝の長さ t_1 ~ t_6 で表現されるとする．

上記の仮定の場合，ある座位 h における図 1B の系統樹の尤度 $L(\Theta|D_h)$は次のように表現される．

$$(1)\quad L(\theta|D_h) = \sum_{x=A,T,G,C} \pi_x \sum_{i=A,T,G,C} p^5_{xi}(t_5)p^1_{ip}(t_1)p^2_{iq}(t_2) \times \sum_{j=A,T,G,C} p^6_{xj}(t_6)p^3_{jr}(t_3)p^4_{js}(t_4)$$

π_a は taxon 1 ~ 4 の実配列データより推定される塩基 a の出現頻度である．$p^k_{bc}(t_k)$は k 番目の枝t_kにおける塩基 b から塩基 c への遷移確率を示し，b から c への瞬間置換速度を各要素にもつ 4×4 の速度行列Q^k に基づき計算される．

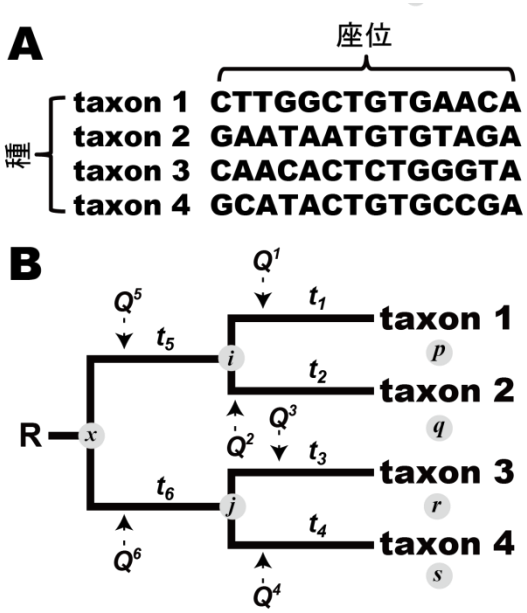

図 1 (A) 4 種からなる塩基配列データ，(B) 有根系統樹

ここで，NH モデルでは系統樹の枝ごとに異なる置換モデルを適用するため，各枝における置換モデルの瞬間置換速度は異なるパラメータによって表現される(図 1B)．例として，NHML で用いられる置換モデルである GG95 モデル[13]の速度行列を以下に示す．

from \ to	A	T	C	G
A	$-(1+\alpha\beta^k)$	$\frac{1-\beta^k}{2}$	$\alpha\frac{1-\beta^k}{2}$	$\frac{\beta^k}{2}$
T	$\frac{\beta^k}{2}$	$-(1+\alpha\beta^k)$	$\frac{1-\beta^k}{2}$	$\alpha\frac{\beta^k}{2}$
C	$\alpha\frac{\beta^k}{2}$	$\frac{1-\beta^k}{2}$	$-\{1+\alpha(1-\beta^k)\}$	$\frac{\beta^k}{2}$
G	$\frac{\beta^k}{2}$	$\alpha\frac{1-\beta^k}{2}$	$\frac{1-\beta^k}{2}$	$-\{1+\alpha(1-\beta^k)\}$

図 2 GG95 モデルの速度行列

図 2 において，α はプリンあるいはピリミジン間での置換(A⇔G／C⇔T)とプリン - ピリミジン間の置換(A⇔T／A⇔C／G⇔T／G⇔C)に対する相対比を示す．また，GG95 モデルでは瞬間置換速度を表すモデルパラメータのうち，塩基 G + C の出現頻度(= β)が各枝で異なることを許容するため，異なる枝に当てはめられた置換モデルの瞬間置換速度は異なる β の値によって決定されることとなる．

従って，GG95 モデルに基づき図 1B の系統樹の尤度を計算する場合，各枝での遷移確率はそれぞれ置換モデル $Q^1 \sim Q^6$に基づき以下のように求められる．

(2) $P^k(t_k)$

$$= U\ diag\left\{\exp\left(\lambda_1^k t_k\right), \exp\left(\lambda_2^k t_k\right), \exp\left(\lambda_3^k t_k\right), \exp\left(\lambda_4^k t_k\right)\right\} U^{-1}$$

λ はQ^kの固有値であり，非特異行列 U の列あるいはその逆行列 U^{-1} の行は，それぞれその固有値に対応するQ^kの右および左固有ベクトルである．なお，各座位での塩基置換は他の座位とは独立に起こると仮定するため，最終的な系統樹の尤度 $L(\Theta|D)$は各座位の尤度を総乗することで求められる．

(3) $$L(\theta|D) = \prod_{h=1}^{M} L(\theta|D_h)$$

2.2 尤度の最大化とパラメータの最尤推定

式(1)で示されるように，系統樹の尤度は各枝長および置換モデルのパラメータ(Θ)の関数として表現される．従って，ある系統樹の尤度を計算する場合，系統樹の尤度を最大化するパラメータの最尤推定量を求める必要がある．この最尤推定には，一般的にニュートン-ラフソン法(以下 NR 法)による反復アルゴリズムが用いられる．図 3 に，NR 法のアルゴリズムを疑似コードとして示す．

このアルゴリズムでは，最初にランダムに決められた各パラメータ Θ の初期値より初期尤度が計算される(図 3 の①)．なお，系統樹推測では尤度は極端に小さな値となるため，一般的に 10 を底として対数をとった値が計算される．次に，②で示される繰り返し文によって各パラメータの最尤推定量を反復的に求める．まず，ある枝長またはモデルパラメータのうち一つのパラメータ θ に注目し，他のすべてのパラメータの値を固定することで，該当するパラメータのみによる尤度関数の 1 階・2 階微分を解析的に計算する(③)．これにより，尤度関数を 2 階までのテイラー展開で近似する．NR 法ではこの計算をパラメータごとに繰り返す．なお，尤度は座位ごとに計算されるので，尤度関数の 1 階･2 階微分もまた座位ごとに計算される(④)．ここで，配列データの座位数を M とすると，全ての微分結果を求めるためには，$9 * 4^2 * M$回の浮動小数点演算が必要である．次にテイラー展開の計算より各パラメータの更新値を求め(⑤)，それらをもとに系統樹の対数尤度を再計算する(⑥)．NR 法による反復アルゴリズムでは現在点で求めた対数尤度と 1 つ前の点で求めた対数尤度の差を取り(⑦)，対数尤度差が限りなく小さな値にまで収束したか否かでアルゴリズムの終了判定が行われる(②)．対数尤度差が極めて小さな値をとり，系統樹の対数尤度が一定の値に収束したと判断された場合，現在点における対数尤度および各パラメータの値が最大対数尤度および最尤推定量として返される(⑨)．なお、1 回の繰り返しで計算された 1 階 2 階微分結果は、繰り返しが進むたび，座位ごとの要素に 0 が代入され値が初期化される（⑧）．

表 1 では，66 種， 2,500 座位および 130 種，2,500 座位の配列データそれぞれに基づく系統樹の尤度計算において，「NR 法反復アルゴリズムが全体の計算時間に占める割合」と「各パートの実行時間占有率」を示す．表 1 の示す通り，解析する配列データに関わらず，NR 法に費やされる計算時間は総計算時間の 98%以上を占めている．さらに，NR 法全体の計算時間に対する各パート（図 3 の①～⑨）の占有率を調べると，1 階 2 階微分計算（図 3 の③④）の時間が全体の 90%以上を占め，残りの時間の殆どは微分結果の初期化（図 3 の⑧）に費やされることが分かる．従って，3 章より NR 法の並列化について検討する．

```
Newton-Raphson method ()
 calculate initial log likelihood (lnL) from initial values of branch lengths and model parameters - ①
 WHILE first step or diff lnL < ε - ②
  FOR number of parameters to be estimated - ③
   FOR number of sites - ④
    calculate 1st and 2nd derivatives of site-lnL with respect to a parameter (θ) to be optimized.
   ENDFOR
  ENDFOR
  update parameters - ⑤
  calculate current lnL from updated parameters - ⑥
  IF first step THEN
   initialization of diff lnL
  ELSE THEN
   calculate current lnL - previous lnL (diff lnL)      } ⑦
  reset 1st and 2nd derivatives - ⑧
 ENDWHILE
 RETURN current lnL as maximum lnL - ⑨
```

図 3　NR 法反復アルゴリズムの疑似コード

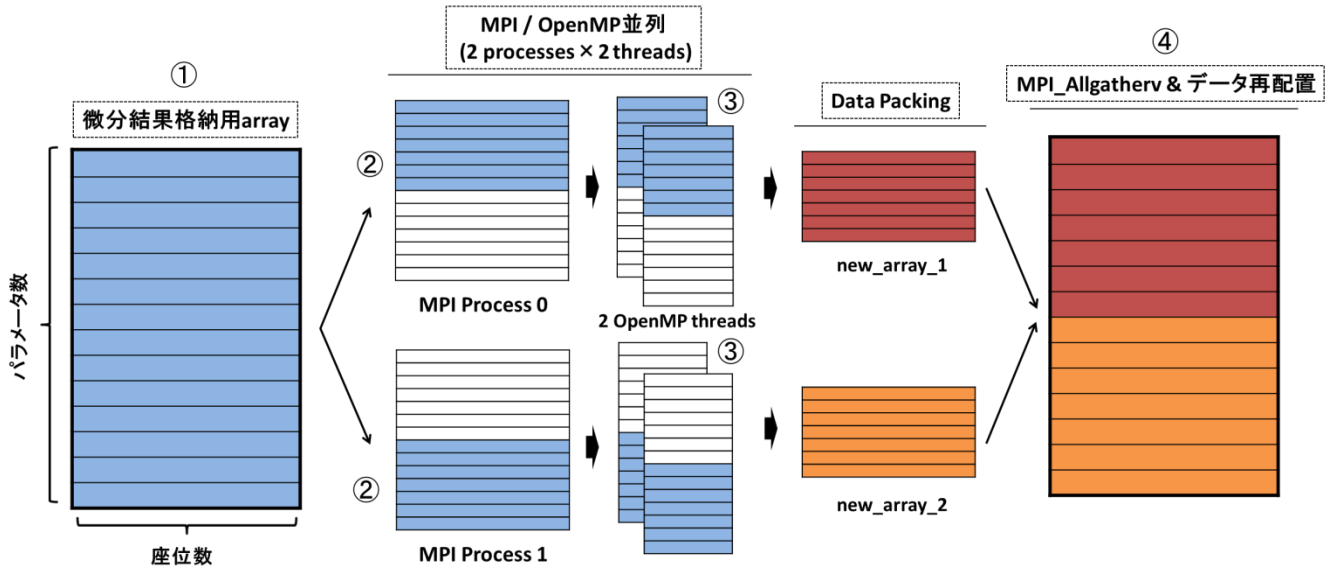

図 4　NR 法の MPI/OpenMP ハイブリッド並列化

	66 種, 2,500 座位	130 種, 2,500 座位
NR 法/総時間 (%)	98.87	98.86
各パート占有率		
③ + ④ (%)	94.4	92.0
⑧ (%)	5.4	7.9

表 1　NR 法における計算時間内訳

3. MPI/OpenMP による系統樹推測の並列化

3.1 NR 法による尤度計算の並列化

系統解析プログラムの並列処理化にあたり，NR 法による反復アルゴリズムのうち，尤度関数の 1 階・2 階微分を求めるための 2 つの for 文に注目した(図 3 の③と④). 異なる置換モデルを系統樹の各枝に適用した場合，最尤推定すべきパラメータ数がモデルの数だけ増加するため, ③の for 文の繰り返し数もまた増大することとなる．また，系統解析で使用される配列データの座位数は一般的に数千～数万であるため, ④の for 文の繰り返し数もまた非常に大きい．さらに，パラメータごと，および座位ごとの微分計算は独立に行うことが出来る．そこで，本研究では③の for 文を MPI，④の for 文を OpenMP によってそれぞれ並列化し，これらを組み合わせたハイブリッド並列を施すことでプログラムの効率的な高速化を図った．このハイブリッド並列では，1 本の系統樹の尤度最大化およびパラメータの最尤推定は下記のように並列計算される．

1). 尤度関数の 1 階・2 階微分の計算をパラメータごとに MPI の各プロセスに分割.
2). プロセスに分けられた微分計算を，さらに座位ごとに OpenMP の各スレッドに分割.

尤度関数の 1 階 2 階微分結果はパラメータ数を行，座位数を列とする array データにそれぞれ格納される（図 4 の①）．GG95 モデルの場合，NR 法で最尤推定されるパラメータおよびパラメータ数はそれぞれ以下の 3 つに分類される．

a)　根における G+C 含量(パラメータ数=1)
b)　各枝の長さ(パラメータ数=2N−1，N は種数)
c)　各枝における G+C 含量(パラメータ数=2N−1)

また，微分計算結果は double 型で表される．従って，微分結果を格納する array データサイズ S は種数 N 座位数 M に対し次のように表される．

(4)　$$S = M * \{2(2N-1)+1\} * 8_{BYTES}$$

なお，1 階微分値および 2 階微分値はそれぞれ同じサイズをもつ異なる array に格納されることに注意する．

図 4 では，2 つの MPI プロセスと、それらを 2 つのスレッドに分割したハイブリッド並列例を示す．それぞれの MPI プロセスは array の全要素のうち自身に割り当てられたパラメータに該当する要素についてのみ微分計算結果を格納する（図 4 の②; 便宜的に array は 1 つのみ表示）．さらに，各プロセスにおいて座位ごとの微分計算は OpenMP スレッドに分配され, 並列に計算される（図 4 の③). なお，反復アルゴリズムでは，各パラメータおよび対数尤度の値を更新するために全てのパラメータについての 1 階・2 階微分の情報が必要である(図 3 の⑤⑥)．従って，2 つの for 文の並列処理が全て終了した後, プロセス間で同期を取り，それぞれの MPI プロセスが全ての 1 階・2 階微分の情報を共有するために関数 MPI_Allgatherv を用いて通信を行わせる（図 4 の④）．なお，1 階微分結果および 2 階微分結果は別々の aray に格納されるため，NR 法の 1 ステップでは計 2 回の Allgatherv 通信が行われる．通信終了後，各プロセスにおいて対数尤度の再計算を行い，系統樹の尤度が収束するまでアルゴリズムを反復する．

3.2 Data Packing による通信データ削減

図 4 のとおり，微分計算の MPI/OpenMP 並列化では，各 MPI プロセスは自身に割り振られたパラメータ分の微分計算のみ行う．従って，それぞれの MPI プロセスのもつ array データでは，そのプロセスが担当するパラメータ以外のパラメータに対しては空の値(= 0)が格納される．ここで，空の要素を含んだ array データをそのまま MPI_Allgatherv 通信に用いると，不要なデータが通信されることになり，通信データサイズの不必要な増加を招く可能性がある．そこで，以下の手順により array データの packing および通信後データの再配置を行うことで，通信データを効率的に削減した．

i). それぞれのプロセスが「何番目のパラメータの微分計算を担当するか」を記録する．
ii). 各プロセスで，担当するパラメータ数×座位数分の要素を持つ array データを calloc 関数により新たに生成し，パラメータごとの微分計算結果をこれに順に格納する (図 4，new_array_1 & new_array_2).
iii). 関数 MPI_Allgatherv により，new_array データを，全パラメータ数×座位数分の要素をもつ元の arrayデータに集約する．この際，i)の手順で記録したパラメータ番号を参照し，各プロセスより集められた微分計算結果を正しい順に配置する．

この data packing により 1 回の Allgatherv 通信で削減されるデータサイズ S_1 は，全体のパラメータ数を P，各プロセスの担当するパラメータ数を p，座位数を M とすると，以下のように計算できる．

$$S_1 = \sum_{k=1}^{num\_procs} 8_{Byte} \times (P - p_k) \times M \quad (5)$$

3.3 複数系統樹の並列的尤度計算

3.1 および 3.2 節では「1 本の系統樹に対する尤度計算」の並列化について述べたが，実際の系統解析では，膨大な樹形空間より最尤系統樹を選択するため，複数本の系統樹の尤度計算を行い，これらを比較する必要がある．ここで，異なる樹形をもつ系統樹の尤度計算は独立に行えるため，複数系統樹の尤度計算もまた並列的に処理することが可能である．本研究では複数系統樹の並列的尤度計算を行うため，全ての MPI プロセスを管理するコミュニケータ(MPI_COMM_WORLD)を複数のサブコミュニケータに分割した．これらのサブコミュニケータに MPI プロセスおよび OpenMP スレッドを等分し，コミュニケータごとに異なる樹形をもつ系統樹の尤度計算をそれぞれ割り当てた．1 つのサブコミュニケータ内では分配された MPI プロセスおよび OpenMP スレッドによって上述した MPI/OpenMP ハイブリッド並列処理が可能であり，さらなる並列化効率が期待できる．

4. 性能評価の問題設定

並列化を行った NHML の性能評価を行うため，塩基配列データを用いて分子系統解析を行った．評価実験に使用する塩基配列データは，図 5A に示すモデル系統樹に基づき，モンテカルロシミュレーション法[14]を用いて生成した．このシミュレーションでは，系統樹の根(R)にて祖先配列が生成され，系統樹の分岐，各枝長および置換モデルのパラメータに従って末端配列が進化する．この際，NHML が実装する GG95 モデルの性質を考慮し，配列中の G+C 含量を示すモデルパラメータ値を各系統で変化させることで，最終的に G+C 含量が生物種間で不均一な配列データを得た．実験では，系統樹の根において G+C 含量 = 50%の設定で塩基配列を生成し，モデル系統樹の二つの節(図 5A：赤矢印)でのみ，G+C 含量 = 90%の設定で以後の配列を進化させた．結果，最終的に生成される末端配列のうち，taxa 1 & 2 および taxa 63 & 64 のみ，他とは異なる進化プロセスを経験することで極端に大きな G+C 含量を示すようシミュレーションを行った．なお，taxa 1 & 2 および taxa 63 & 64 に至る末端枝(図 5：赤線)では，G+C 含量のバイアスを反映させるため枝長＝置換速度が他の枝よりも 10 倍ほど長くなるよう設定した．図 5A では系統関係のみを phylogram として示す．

上記の設定より生成された塩基配列データを Homogeneous 置換モデル（系統間での進化プロセス変化を許容しない単純モデル）を用いた系統解析法に供した場合，図 5B に示すように，本来系統的に離れた taxa 1 & 2 と taxa 63 & 64 が近縁であるかのように誤推測される．そこで，以下の手順に基づき，誤推測された系統樹より真の系統樹(＝モデル系統樹と同じ樹形)を復元するための系統樹探索を行った．

(I) 図 5B で示される樹形を初期樹形とし，初期樹形の尤度計算を行う．
(II) 初期樹形のうち，taxa 63 & 64 からなる部分木(図 5B：赤破線)を刈り取り，系統樹の別の節へと接合することで，異なる樹形を提案する．(部分木剪定・接木法)
(III) 部分木剪定・接木法では，刈り取った部分木を移動する距離（＝移動先の節との間にある内部枝数）を調整し，提案樹形の中に必ず真の系統樹が含まれるようにする．
(IV) 提案樹形それぞれについて尤度計算を行う．
(V) 初期樹形と各提案樹形とで対数尤度比較を行い，最も大きな対数尤度をもつ樹形を最尤系統樹として選択する．

図 5 では 66 種(64 種 + 外群 2 種)からなる系統樹のみ示すが，130 種(128 種+ 外群 2 種)からなるモデル系統樹も用意した．130 種モデル系統樹においても G+C 含量を系統間で変化させることで，taxa 1，2 および taxa 127，128 の末

端配列で G+C 含量バイアスを生じさせた．130 種系統樹についても 66 種系統樹と同様な初期系統樹を用意し，(I) ~ (IV)の系統樹探索により最尤系統樹を推測した．66 種，130 種系統樹から生成される提案樹形はそれぞれ 24 本，48 本である．

実験では，それぞれのモデル系統樹について 2,500 座位からなるシミュレーション配列を生成した．また，66 種系統樹についてはさらに 10,000 座位からなる塩基配列データを生成した．以上のシミュレーション配列データセットを並列化した NHML に供し，最尤系統樹の推測終了までに要した時間を計測した．

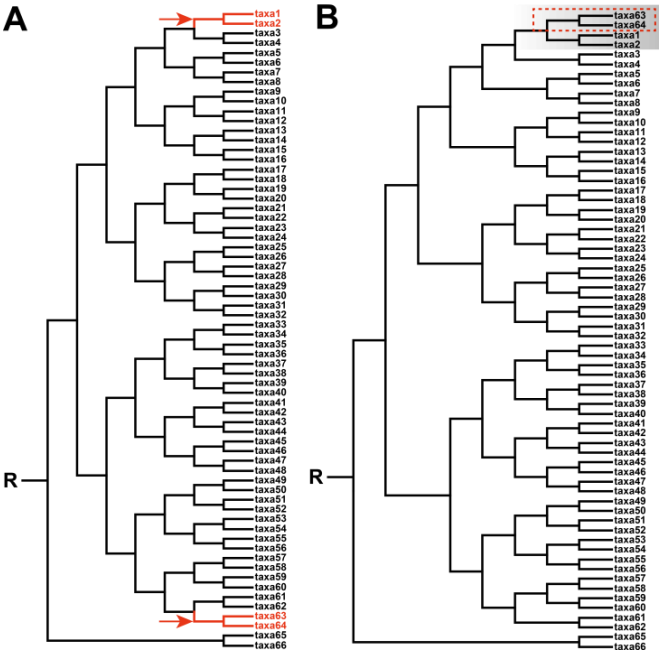

図 5 (A) 66 種モデル系統樹，(B) 初期系統樹

5. 結果と考察

5.1 測定環境

データ解析は T2K-Tsukuba System の最大 64 ノードまでを使用した．計算環境の詳細は表 2 に示す．

なお，ハイブリッド並列処理において， MPI の 1 プロセスは T2K-Tsukuba System の 1 つのソケットに割り当てるものとし，さらにソケット内部の 4 つの CPU コアに 4 つの OpenMP スレッドをそれぞれ割り当てた．従って，1 つの計算ノードでは 4 つの MPI プロセスが各ソケットに割り当てられることで分散メモリ型並列処理を行い，同時にソケット内部では各 MPI プロセスの管理する OpenMP スレッドによる共有メモリ型並列処理が行われる．ジョブ実行時には「numactl –cpunodebind –localalloc」のオプション指定を行った．

CPU	Quad-core AMD Opteron 8356 (2.30GHz) (4 コア x 4 ソケット / ノード)
Memory	DDR2, 667MHz, 2GB x 16 = 32GB (1 ノード)
Network	Infiniband 4xDDR, Mellanox ConectX x 4
Compiler	GCC 4.6.4
MPI Library	MVAPICH2 Ver. 1.7

表 2 測定環境（T2K-Tsukuba）のシステム概要

5.2 NR 法反復アルゴリズム並列化に対する評価

5.2.1 種数の異なる配列データ解析結果

4.1 節で述べた 66 種および 130 種系統樹より生成された 2,500 座位塩基配列データの解析に要する時間を，並列数 16 ～ 256 の場合について調べた．図 6AB より，66 種データおよび 130 種データいずれの解析においても，並列数 256 まで，解析時間の短縮が確認された．図 6C では，16 並列を基準としたそれぞれのデータ解析における並列化効率の変化を示す．結果より，3.1 節で述べた NR 法アルゴリズムのハイブリッド並列化では，並列数 128 までは，いずれの配列データ解析でも，0.5 より大きな並列化効率(efficiency)を維持することが出来た．一方，並列数を 256 まで上げた場合，130 種からなる配列データの解析では継続して efficiency > 0.5 と良好な効率を示したが，66 種データ解析では並列化効率は 0.5 を下回った．

5.2.2 座位数の異なる配列データ解析結果

図 7AB では，66 種データ解析において，座位数を 2,500 から 10,000 へと変化させた場合の総計算時間および並列化効率の変化を示す．実験結果から，座位数が変化した場合でも，並列数に増加に応じて総計算時間の短縮がみられ，種数を変化させた場合と同じく並列数 128 までは efficiency > 0.5 と良好な並列化効率を示した．並列数 256 では 2,500 座位データ解析では efficiency < 0.5 であった一方，10,000 座位データ解析では efficiency > 0.5 と良好な速度向上率を示した．

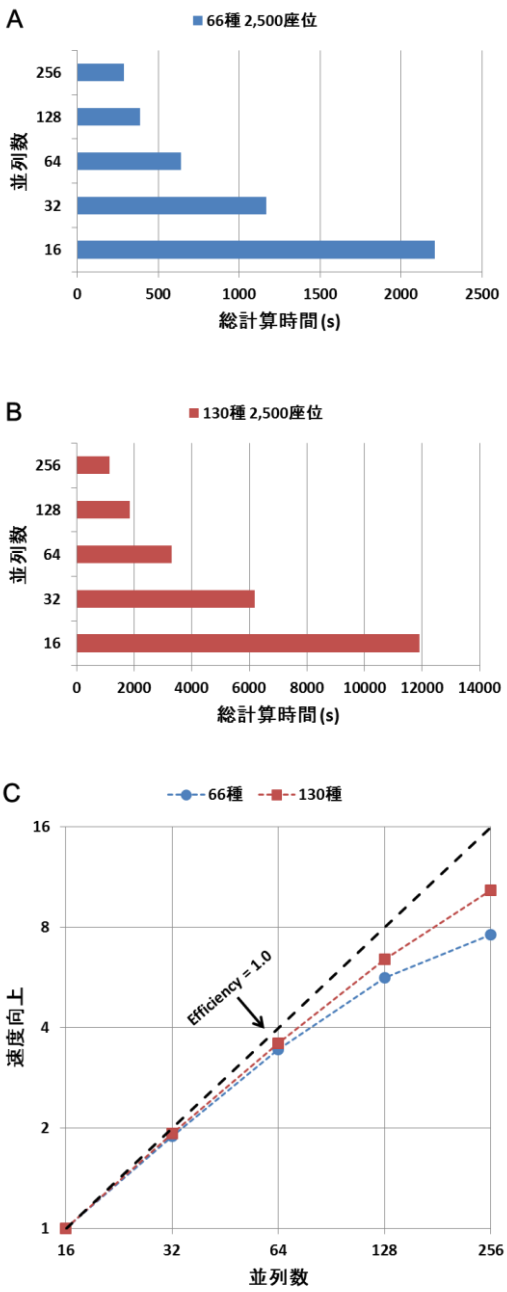

図 6　種数の異なるデータ解析における速度向上の変化. (A) 66 種データ解析における総計算時間の変化. 縦軸は CPU コア数, 横軸は全ての系統樹計算に要した時間(s). (B) 130 種データ解析における総計算時間の変化. (C) 種数の増加に対する速度向上率の変化. 縦軸は 16 コア使用時の計算時間を基準とした速度向上率. 横軸は CPU コア数. 破線は並列化効率＝1.

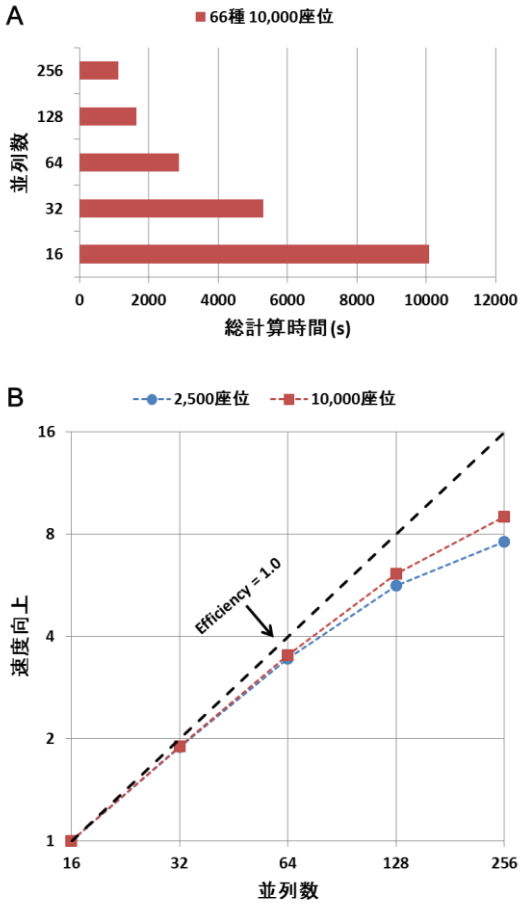

図 7　座位数の異なるデータ解析における速度向上変化. (A) 10,000 座位データ解析における総計算時間の変化. (B) 座位数の増加に対する速度向上率の変化.

5.2.3 Allgatherv 通信負荷の並列化効率への影響

図 6 および図 7 の結果より,「使用するコア数に対しどこまで効率的な速度向上が見込めるか」という問題については, 解析データの種数および座位数, すなわち配列データのサイズが並列化効率に影響することが示唆された. そこで, 種数ならびに座位数の最も小さな 66 種 2,500 座位のデータ解析において, 並列数 256 で並列化効率が大きく減少した原因を調査するため, それぞれの並列数において, 総計算時間に対する MPI_Allgatherv 通信時間および CPU 計算時間の割合を測定した. 図 8 の示す通り, 並列数が上がるにつれ, MPI_Allgatherv 通信に費やされる時間は増大し, さらに総計算時間に対する MPI 通信時間の割合も上昇した. 並列数 256 では総計算時間のおよそ 50% を MPI_Allgatherv 通信が占める結果となった.

NR 法アルゴリズムのハイブリッド並列処理において, 1 つの MPI プロセスが扱う array データ量 S_p, 演算量 C_p は, 配列データの座位数を M, プロセスが微分計算を担当するパラメータ数を p とすると, 以下のように表すことが出来る.

(6)　$$S_p = p \times M \times 2$$

(7) $$C_p = p \times (9 \times 4^2 \times M) \times 2$$

また，MPI_Allgatherv 通信では，あるプロセスは自身の担当したパラメータ以外の全てのパラメータについて微分結果を受信する必要がある．従って，1 プロセスにおける MPI_Allgatherv 通信量 T_p は，微分結果格納用 array 全体のサイズを S（式 4 を参照）とすると，以下のように表せる．

(8) $$T_p = 2 \times (S - S_p)$$

ここで，各プロセスの担当するパラメータ数 p は，並列数が大きくなり MPI プロセス数が増大するにつれ減少する．従って，並列数（MPI プロセス数）を上げた場合，データ量 S_p および演算量 C_p は減少する一方，通信量 T_p は増大することが分かる．

以上より，並列数をある一定数より大きくした場合，並列化による恩恵（＝各プロセスにおける CPU 演算の減少）を，MPI 通信コストが上回ることで，全体の並列化効率が低下することが予想される．

一方，データ量 S_p および演算量 C_p は P ならびに M に比例し，P は配列データの種数に比例するため（3.1 節参照），各プロセスの扱う演算量は種数および座位数に比例する．そのため，良好な並列効率を得られる並列数の上限は，解析するデータサイズに影響されることが示唆される．従って，より大きな種数・座位数をもつ配列データを解析する場合には，並列数を大きく上げたとしても，良好な並列化効率を維持できると期待できる．実際に，130 種 2,500 座位データおよび 66 種 10,000 座位データの解析では，並列数 256 においても efficiency > 0.5 と良好な並列化効率が維持されており（図 6C, 図 7B），256 より並列数を上げた場合でもさらなる速度向上が望まれる．

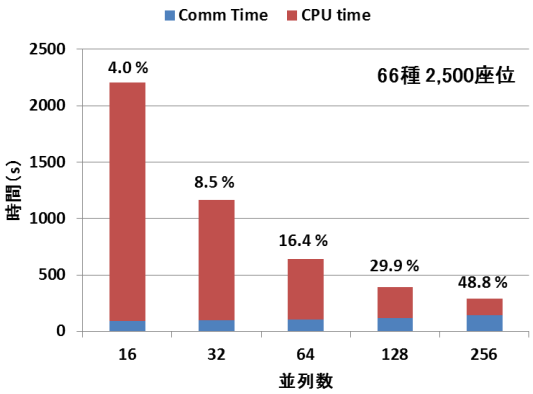

図 8　66 種 2,500 座位データ解析における，総計算時間に対する MPI_Allgatherv 通信時間および CPU 時間の割合の変化．各棒グラフ上の数字は MPI_Allgatherv 通信時間の総計算時間に対する割合を示す．

5.3 Flat MPI 版プログラムとの性能比較

OpenMP を使用せず，MPI のみで NR 法の並列化を行った場合（Flat MPI）と，ハイブリッド並列化を行った場合とで，速度向上や並列化効率にどの程度違いが生じるかを調べた．図 9 では，66 種，10,000 座位の配列データ解析において，並列数を 256 としたときの，ハイブリッド並列版プログラムおよび Flat MPI 並列版プログラムの総解析時間をそれぞれ示す．結果より，総計算時間（CPU time + Comm time）をみると，Flat MPI 版プログラムはハイブリッド版プログラムより，32%程度性能が落ちることが分かった．さらに，Flat MPI 版ではハイブリッド版に比べ CPU による計算時間が 77%減少したが，一方で MPI_Allgatherv 通信に要する時間（プロセス間の同期およびデータ送受信）は 3.5 倍ほど増大した．また，Flat MPI 版プログラムでの通信時間は全体の 91%程度を占めており，通信時間の割合が 43%程度のハイブリッド並列版に比べ，MPI 通信に多大な負荷が掛かった．

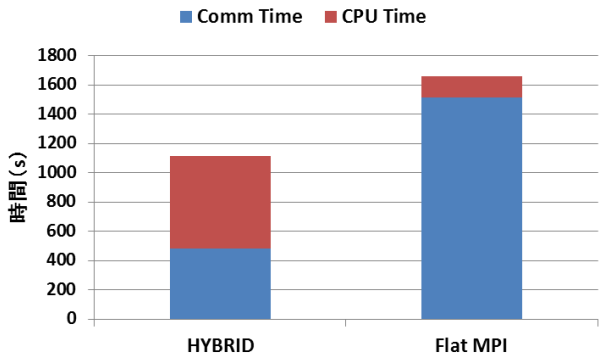

図 9　ハイブリッド並列と Flat MPI 並列の性能比較

Flat MPI 並列では，ハイブリッド並列に比べより多くの MPI プロセスを「パラメータごとの 1 階 2 階微分計算」に割り当てられるため，1 つのプロセスの取り扱う演算量および CPU 演算時間は減少する．特に，ハイブリッド並列版では座位ごとの微分計算を OpenMP によって並列処理しているにも関わらず，Flat MPI 版に比べ CPU 演算に多くの時間を要した．従って，各パラメータに対する微分計算の MPI 並列化は，各座位に対する微分計算の OpenMP 並列化に比べ，CPU 演算に対する速度向上効果が大きいと示唆される．しかしながら，5.2.3 節で述べた通り，MPI プロセス数が大きくなるにつれ，Allgatherv 通信にて各プロセスの取り扱うデータ量および MPI 通信時間は増大する．図 9 より，「ハイブリッド/Flat MPI 間の MPI 通信時間の差」は，「ハイブリッド/Flat MPI 間の CPU 演算時間の差」に比べ遥かに大きい．このことから，Flat MPI 版では，「CPU 演算の減少」を「MPI 通信負荷の増大」が大きく上回ったことで，総計算時間がハイブリッド版に比べ増大したと推測される．

一方で，ハイブリッド並列版では，同じ並列数で MPI プロセス数を減らすことで通信コストを下げる代わりに，通信の必要ない OpenMP によるスレッド並列を併用することで, Flat MPI 版よりも良好な速度向上を達成した. 従って，より多くの CPU コアを使用して良好な並列化効率を得るためには，ハイブリッド並列によるプログラムの高速化が推奨される.

5.4 複数系統樹の並列的尤度計算に対する評価

図 6，図 7 および 5.2.3 節で示されたとおり，種数・座位数によって上限は変化するが，1 本の系統樹の尤度計算に対する並列化では，ある一定の並列数以上では効率的な速度向上が見込めないことが予想される．そこで，3.3 節で述べたように，MPI コミュニケータを複数のサブコミュニケータに分割し，それぞれのコミュニケータに樹形の異なる系統樹の尤度計算を割り当てることで，複数系統樹の尤度計算を並列的に行わせた．コミュニケータの分割方針については，①各サブコミュニケータに 1 つの計算ノード(4 プロセス×4 スレッド＝16 CPU コア)を割り当てる，②2 つの計算ノード(8 プロセス×4 スレッド＝32 CPU コア)を割り当てる，③4 つの計算ノード(16 プロセス×4 スレッド＝64 CPU コア)を割り当てる，3 つのパターンを用意した.

実験では，130 種，2,500 座位からなる配列データを用いた解析を評価対象とした．130 種配列データの解析の場合，4 章で述べた部分木剪定・接木法によって初期系統樹から生成される提案樹形数は 48 本であり, これらの樹形の尤度計算を①②③の異なるパターンで生成されたサブコミュニケータ群に分配すると，各コミュニケータが尤度計算を担当する樹形数は表 3 に示す通りになる.

	並列数		
	256	512	1024
①(4 x 4)	3		
②(8 x 4)	6	3	
③(16 x 4)	12	6	3

表 3　コミュニケータ分割パターンと並列数に対する，各コミュニケータの担当樹形数

図 10 では，総並列数 256 ~ 1024 に対し，全ての提案樹形の尤度計算に要した時間を示す．並列数 256 では，MPI コミュニケータの分割を行わず, 図 6, 7 と同じく 1 本の系統樹の尤度計算に全ての MPI プロセスおよび OpenMP スレッドを割り当てた場合の計算時間も併せて示す. 結果より，並列数 256 では，いずれの分割パターンによる解析においても，全ての計算資源を 1 本の系統樹の尤度計算に割り当てる場合（図 10 青）に比べ，解析に要する時間はより短く済んだ．図 6 および図 7 で示唆されたように，並列数が増加するにつれ，1 本の系統樹に対する尤度計算の並列化効率は徐々に減少する傾向にある．従って，多くの MPI プロセスおよび OpenMP スレッドを使用する場合，それらを分割して異なる樹形の尤度計算に割り当てる方が，全ての計算資源を 1 本の系統樹の尤度計算に費やす場合に比べ，最尤系統樹推測をより高速に行うことが出来る．なお，並列数 256 では②の分割パターンによる解析で最も良い速度向上が得られた. これは, 複数の MPI プロセスおよび OpenMP スレッドを 1 本の系統樹の尤度計算に割り当てたことによる速度向上効果（図 6）と，複数のサブコミュニケータを用いた並列的尤度計算による速度向上効果が，最も効率的に働いた結果であると思われる．また，並列数 256 において分割パターン①②③の間で速度向上に若干の差がみられたことから，ある一定の並列数のもとで最適な速度向上効果を得るためには，MPI コミュニケータの分割パターンをデータに応じて適宜調整する必要があると示唆される.

1 つのサブコミュニケータに多くの MPI プロセスおよび OpenMP スレッドを割り当てる場合，限られた並列数では各コミュニケータが評価すべき系統樹数が多くなってしまう（表 3）．しかしながら，この問題は，全体の並列数を上げ，サブコミュニケータ数を増やすことで解決することが可能である．実際に，②および③の分割パターンでは，並列数を 512，1024 と増やした場合でも計算時間のさらなる短縮がみられ, パターン③では並列数 1024 において並列数 256 に比べ 3.9 倍の高速化が認められた．最終的に，複数系統樹の並列的尤度計算も含めた結果では，130 種 2,500 座位のデータ解析について並列数 16(図 6)と比べ 40.1 倍の高速化が達成できた.

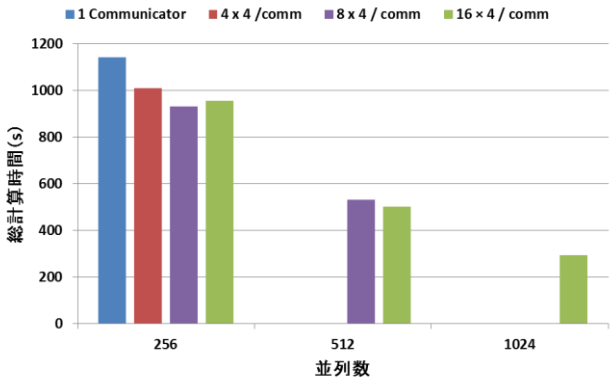

図 10 異なる MPI コミュニケータ分割パターンにおける複数系統樹の並列的尤度計算．縦軸は総計算時間，横軸は並列数を示す.

6. 関連研究

最尤法に基づく系統解析プログラムのハイブリッド並列化については，Pfeiffer and Stamatakis．2010[9]が関連研

究として挙げられる．Pfeiffer and Stamatakis．2010 では複数本の系統樹に対する尤度計算を各 MPI プロセスに分配し，それぞれの系統樹に対する尤度計算については，PTHREADS[15]を用い座位ごとの計算をスレッド並列化するという手法が採られている．一方，本研究では MPI コミュニケータ分割により，複数系統樹の尤度計算を並列的に行うと同時に，NR 法アルゴリズムの並列化により 1 本の系統樹の尤度計算についても MPI/OpenMP ハイブリッド並列化を施すことで，最尤系統樹推測の更なる高速化を実現した．特に，図 10 で示した通り，本研究の手法では，解析するデータサイズや評価すべき提案樹形の数によって，サブコミュニケータ数や各コミュニケータに与える MPI プロセスおよび OpenMP スレッドの数を柔軟に調整することが可能である．この工夫によって，より多くの計算資源（CPU コア）を，効率的に運用することが可能であろう．

7. まとめ

本論文では，NH モデルのひとつである GG95 モデルを実装した系統解析プログラム NHML を対象とし，1 本の系統樹の尤度計算に対する NR 法反復アルゴリズムのハイブリッド並列化を行った．さらに，MPI コミュニケータの分割により，複数本の系統樹に対する尤度計算を並列に行わせた．

NHML を用いた実配列データの系統解析では，一般的に数百種，数千 ~ 数万程度の塩基配列データが用いられる[12]. 本論文で行った性能評価では, 66 および 130 種, 2,500 および 10,000 座位と，種数・座位数の異なる配列データを用いて系統解析を行ったが，いずれの解析においても，プログラムの並列化効率はほぼ一定に保たれた(図 6,7)．特に，本研究の性能評価実験より，ハイブリッド版プログラムは，解析に供すデータの種数または座位数が大きくなるほど，より大きな並列数においても良好な速度向上率を維持することが示唆された．従って，本研究において開発されたプログラムは，大規模配列データに基づく実系統解析の要求に十分に応えるものと期待できる．

一方，N 種からなる配列データに対して考えうる有根系統樹の樹形数は$\{(2N-3)\}!/\{2^{N-2}(N-2)!\}$であるため，大規模配列データの系統解析では，最尤系統樹を探索するため膨大な数の提案樹形に対し尤度計算を行う必要がある．ここで図 10 に示す通り，MPI コミュニケータの分割による複数系統樹の並列的尤度計算では, 並列数 1000 以上においても良好な並列化効率が保たれた．従って，大規模な最尤系統樹探索であっても豊富な計算資源(CPU コア)を用いることで対処することが可能である．

本研究では今後の展望として，多種多様な生物群より得られた大規模塩基配列データをもとに実配列解析用データセットを作成し, 並列化された NHML を用い高性能並列計算機上で分子系統解析を行うことで，真核生物の根の探索[6]や海洋性光合成細菌の多様性解明[16]，γ型プロテオバクテリアにおける昆虫共生菌の起源解明[17]など，複雑な生物進化の解明に挑む．

謝辞　本研究を進めるにあたり，有益なご意見を多数いただきました，筑波大学システム情報工学研究科の児玉祐悦教授，東京大学の塙敏博准教授，ならびに，理化学研究所計算科学研究機構の辻美和子博士に深く感謝いたします．本研究の一部は，筑波大学計算科学研究センターT2K Tsukuba System 一般利用プログラム「NONHOMO」および学際共同利用プログラム「XMP」によるものです．

参考文献

1). Lockhart, P. J., Steel, M. A,, Hendy, M. D. and Penny, D. Recovering evolutionary trees under a more realistic model of sequence evolution. *Mol. Biol. Evol*. 11:605-612. (1994).

2). Ho, S. Y. W. and Jermiin, L. S. Tracing the decay of the historical signal in biological sequence data. *Syst. Biol*. 53:623-637.

3). Leigh, J. W., Susko, E., Baumgartner, M., and Roger, A. J. (2008) Testing congruence in phylogenomic analysis. *Syst. Biol*. 57:104-115. (2004).

4). Phillips MJ, Delsuc F, Penny D. Genome-scale phylogeny and the detection of systematic biases. *Mol. Biol. Evol.* 21:1455-1458. (2004).

5). Dutheil, J. and Boussau, B. Non-homogeneous models of sequence evolution in the Bio++ suite of libraries and programs. *BMC Evol. Biol*. 8:255-255. (2008).

6). Williams, T. A., Foster, P. G., Nye, T. M., Cox, C. J., & Embley, T. M. A congruent phylogenomic signal places eukaryotes within the Archaea. *Proc. Roy. Soc. B: Biological Sciences*, 279(1749), 4870-4879. (2012).

7). Ishikawa, S. A., Inagaki, Y., and Hashimoto, T. RY-Coding and Non-Homogeneous Models Can Ameliorate the Maximum-Likelihood Inferences From Nucleotide Sequence Data with Parallel Compositional Heterogeneity. Evolutionary *Bioinformatics* Online, 8, 357. (2012).

8). Morgan, C. C., Foster, P. G., Webb, A. E., Pisani, D., McInerney, J. O., & O'Connell, M. J. Heterogeneous models place the root of the placental mammal phylogeny. *Mol. Biol. Evol*, 30(9), 2145-2156. (2013).

9). Pfeiffer, W., & Stamatakis, A. Hybrid MPI/Pthreads parallelization of the RAxML phylogenetics code. *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), IEEE International Symposium on*. pp. 1-8. IEEE. (2010).

10). Pratas, F., Trancoso, P., Stamatakis, A., & Sousa, L. Fine-grain parallelism using Multi-core, Cell/BE, and GPU systems: Accelerating the phylogenetic likelihood function. *Parallel*

Processing, *ICPP'09*. *IEEE International Symposium*. pp. 9-17. IEEE. (2009).

11). Galtier, N., and Gouy, M. Inferring pattern and process: maximum-likelihood implementation of a nonhomogeneous model of DNA sequence evolution for phylogenetic analysis. *Mol. Biol. Evol.*15:871-879. (1998).

12). Escobar, J. S., Glemin, S., & Galtier, N. GC-biased gene conversion impacts ribosomal DNA evolution in vertebrates, angiosperms, and other eukaryotes. *Mol. Biol. Evol.* 28(9), 2561-2575. (2011).

13). Galtier, N., & GouY, M. A. N. O. L. O. Inferring phylogenies from DNA sequences of unequal base compositions. *Proc. Natl. Acad. Sci. U.S.A*, 92(24), 11317-11321. (1995).

14). Fletcher, W., & Yang, Z. INDELible: a flexible simulator of biological sequence evolution. *Mol. Biol. Evol*, 26(8), 1879-1888. (2009).

15). Lewis, B., & Berg, D. J. Multithreaded programming with Pthreads . *Sun Microsystems Press*. Vol. 2550. (1998).

16). Szollsi, G. J., Boussau, B., Abby, S. S., Tannier, E., & Daubin, V. Phylogenetic modeling of lateral gene transfer reconstructs the pattern and relative timing of speciations. *Proc. Natl. Acad. Sci. U.S.A*, 109(43), 17513-17518. (2012).

17). Williams, K. P., Gillespie, J. J., Sobral, B. W., Nordberg, E. K., Snyder, E. E., Shallom, J. M., & Dickerman, A. W. Phylogeny of gammaproteobacteria. *J. Bacteriol*, 192(9), 2305-2314. (2010).