G-10

ゲノムマッピング高速化手法の提案

Proposal of Genome Mapping Fast Method

松井 大樹 †　大沢勇統 †　髙橋 篤 ‡　大星 直樹 †
Hiroki Matsui　Yuto Ohzawa　Atsushi Takahashi　Naoki Ohboshi

1. はじめに

近年，次世代シークエンサーの登場により，ヒトをはじめとする様々な生物の DNA 情報の取得および利用が可能になり，遺伝子解析研究が急速に発展してきた．これにより，ゲノム情報を医療，創薬等に応用しようとする動きが高まってきている．ゲノム解析において，読み取った短い塩基配列の断片について，既に特定されているゲノム配列（リファレンス配列）のどの部分に相当するかを決定するマッピングと呼ばれる処理が行われる．遺伝子の変異解析，特に一塩基多型解析などの一塩基単位での変異を見つけるような高い精度を必要とする解析においては，正しいマッピングを行うことが重要である．マッピングを行うツールは複数ある．いずれのマッピングツールを用いた場合でも，扱うデータのサイズが大きいため，処理に膨大な時間が必要となる．現状，処理時間の短縮はマシンのスケールアップに頼っている．しかし，マシンのスケールアップだけでは処理時間の短縮に限界がある．そこで，本稿ではマッピング処理を並列化し，複数台のマシンで構築した PC クラスタ上で動作させることで，処理速度を向上させ，処理にかかる時間を短縮することを目的とした．

速度向上率を検証するために，最大ノード数を 8 台とし，ノード数 1, 2, 4, 8 台でそれぞれマッピング処理を行い，各ノード数における処理時間を測定し，ノード数 1 台の場合と比較した速度向上率を算出した．

2. ゲノム解析

本章ではゲノム解析とゲノム解析におけるマッピングについて説明する．

2.1 ゲノム解析

ゲノムとは生物の DNA に存在する遺伝子情報の集合である．現在では生物の全 DNA の塩基配列を解析することが比較的容易となったため，特定の生物の持つ全 DNA の塩基配列情報をゲノムと呼ぶことが多い．ゲノムは DNA 分子の塩基配列（ATGC のならび）から構成される．地球上の生物はそれぞれ独自のゲノムを持っており，ゲノムの多様性が生物種に多様性を与えている．多様性を示す一つの指標はゲノムの大きさ（総塩基数）である．しかし，生物はゲノムの中に生存に必ずしも必要でない DNA を多数抱えている．そのため，ゲノムサイズが大きいからといって，遺伝子の種類数が多い複雑な生物であるとは限らない．

ゲノム解析では，はじめに次世代シークエンサーを用いて DNA の塩基配列を読み取る．次世代シークエンサーとは生物の DNA 塩基配列のならびを読み取る機器のことである．次世代シークエンサーではゲノムを一本の配列として読み取ることができないため，塩基配列解読の方法として，ゲノム配列をランダムに切断し，断片化する．次に，断片化された配列を復元し，解析対象個体のゲノム配列を得る．得られた配列情報から遺伝子の同定や機能の推定などの解析を行う．

2.2 マッピング

次世代シークエンサーによって読み取られ，断片化された配列を復元する手法の一つにマッピングがある．マッピングとは，ゲノム配列の断片（リード）を参照ゲノム配列と比較し，ゲノムの断片がどの位置にあたるのかを判断してはりつけることで，個体ゲノムの配列を再現する処理のことである．参照ゲノム配列には，ヒトやマウス，ショウジョウバエなどの既に全ゲノム情報が決定された生物の全ゲノム配列を使用する．この処理を行うソフトウェアは複数存在し，Burrows-Wheeler Transform(BWT)[1]アルゴリズムに基づいた処理を行うものが主に用いられている．マッピングを行う代表的なソフトウェアとして BWA[2]，Bowtie2[3]があり，どちらもフリーツールである．それぞれのマッピングツールの特徴を以下に示す．

・BWA
　マッピングするリードの長さが短いショートリード向きのマッピングルールである．BWT をベースとし，ある程度のギャップを許容してマッピングする．1000bp のような長いリードにも対応可能である．FASTQ 形式のファイルを入力データとして，結果を SAM 形式のファイルとして出力する．デフォルトで Indel（塩基の挿入，欠失）の検出を行うことができる．処理はアライメント処理とマッピング処理の 2 ステップに分かれており，アライメント結果からマッピングを行う．

・Bowtie2
　リード長が 50~100bp 程度の短いリードをマッピングするのに用いられるツールである．FASTQ 形式のファイルを入力データとして，結果を SAM 形式のファイルとして出力する．デフォルトで Indel（塩基の挿入，欠失）の検出を行うことができる．

いずれのツールを用いた場合でも扱うデータのサイズが大きいために処理に膨大な時間が必要となる．これはゲノム解析の進展におけるボトルネックの一つである．

3. システム概要

3.1 使用ツール

本稿では，前述のマッピング処理に広く用いられているフリーツールである BWA について並列化を行った．また，マッピングツールの並列化には Message Passing Interface(MPI)のライブラリの一つである OpenMPI[4]を使用した．

†近畿大学大学院総合理工学研究科,
Graduate School of Science and Engineering, Kinki University

‡ 国立循環器病研究センター,
National Cerebral and Cardiovascular Center

3.2 並列計算器

クラスタを構築するためのマシンとしてシングルボードコンピュータの Raspberry Pi 2 Model B を最大 8 台使用した．Raspberry Pi 2 Model B の性能を表 1 に示す．

表 1　Raspberry Pi Model B 1 台あたりの性能

OS	Raspbian
CPU	ARM Cortex-A7 クアッドコア 900MHz 32bit
メモリ	1GB

3.3 ネットワーク構成

ネットワーク構成を図 1 に示す．Raspberry Pi を全て同じサブネットマスクに接続する．ネットワーク接続には 100Mbps Ethernet を使用する図 1 中の Rpi0 をマスタノードとし，接続されている各ノードを制御する．

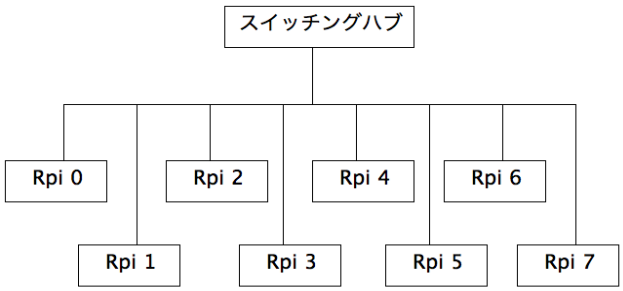

図 1　ネットワーク構成

3.4 実装

並列処理の実装には MPI のライブラリの一つである OpenMPI を使用した．

マッピングツールでは，マッピングを行う前にリファレンス配列に関する情報を持った index ファイルを作成する必要がある，index ファイルは前述の BWT に基づいて作成される．BWA では，マッピングをアライメント処理とマッピング処理の 2 ステップに分けており，断片ゲノム配列データをリファレンス配列にアライメントし，アライメントされたデータから一つのゲノム配列を生成する．BWA の処理において大半をアライメント処理とマッピング処理が占めているため，並列化を index ファイルの作成については行わず，アライメントとマッピング部分についてのみ行った．これにより，プログラム全体の並列化率は約 0.7 となった．実装した並列化の手法について以下に示す．

1. あらかじめ作成した index をすべてのノードにコピーする．
2. マスタノードから各ノードにシークエンスデータを 1 つずつ送信する．
3. 各ノードはシークエンスデータを受け取り，index からシークエンスデータを探索（アライメント）する．
4. 各ノードはアライメント結果からマッピングを行う．
5. マッピング結果を各ノードからマスタノードに送信する．
6. マスタノードは結果を受け取ったノードに次のシークエンスデータを送信し，受け取った結果を同期する．
7. すべてのシークエンスデータに対して 2~6 を行う．

4. 実験

4.1 使用データ

シークエンスデータとして National Center for Biotechnology Information(NCBI)[5] にて公開されている Escherichia coli str. K-12 substr. MG1655(SRR001666) のゲノムデータを使用した．シークエンスデータには読み取られたゲノムの断片の片側のみを読んだ single-end read と両側を読んだ paired-end read が存在する．本稿では，paired-end read の SRR001666_1.fastq, SRR001666_2.fastq（リード数：7,047,668，塩基数：76 塩基/リード）を使用した．使用した FASTQ ファイルの一部を以下に示す．

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=36
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGA
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBI
@SRR001666.3 071112_SLXA-EAS1_s_7:5:1:839:309 length=36
GAATTTCAATACGGGTGACTTTAATCCCCCACGGGT
+SRR001666.3 071112_SLXA-EAS1_s_7:5:1:839:309 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII/
@SRR001666.4 071112_SLXA-EAS1_s_7:5:1:792:346 length=36
GATACCCAGAATACCAAACACCGTCTGCTGCATATC
+SRR001666.4 071112_SLXA-EAS1_s_7:5:1:792:346 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9I6II
@SRR001666.5 071112_SLXA-EAS1_s_7:5:1:765:373 length=36
GGTGTGTTGAGCTTGTCCAGCCTGGTGGATTCGATG
+SRR001666.5 071112_SLXA-EAS1_s_7:5:1:765:373 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII;IIII8II
@SRR001666.6 071112_SLXA-EAS1_s_7:5:1:790:604 length=36
GCCATGTTTATTACCTGGAGAAACTTACAATTTTAT
+SRR001666.6 071112_SLXA-EAS1_s_7:5:1:790:604 length=36
IIIIIIIIIIIIIIIIII5I7IHIIIFIIBIIIIDI
@SRR001666.7 071112_SLXA-EAS1_s_7:5:1:846:612 length=36
GATCCTGATGCAACGTGCCCACCAGCGCCAGTTGGG
+SRR001666.7 071112_SLXA-EAS1_s_7:5:1:846:612 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@SRR001666.8 071112_SLXA-EAS1_s_7:5:1:823:618 length=36
GATTTAGGATTAACCACTGAGTATTTAGAAACCTTC
```

シークエンスデータをマッピングするために用いる参照配列（リファレンス配列）は ecoli_k12_mg1655.fasta（塩基数：約 460 万塩基）を使用した．FASTA ファイルの一部を以下に示す．

```
>gi|556503834|ref|NC_000913.3| Escherichia coli str. K-12 substr. MG1655, complete genome
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC
TTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAA
TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAACATCCATGAAACGCATTAGCACCACC
ATTACCACCACCATCACCATTACCACAGGTAACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAG
CCCGCACCTGACAGTGCGGGCTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAA
GTTCGGCGGTACATCAGTGGCAAATGCAGAACGTTTTCTGCGTGTTGCCGATATTCTGGAAAGCAATGCC
AGGCAGGGGCAGGTGGCCACCGTCCTCTCTGCCCCCGCCAAAATCACCAACCACCTGGTGGCGATGATTG
AAAAAACCATTAGCGGCCAGGATGCTTTACCCAATATCAGCGATGCCGAACGTATTTTTGCCGAACTTTT
GACGGGACTCGCCGCCGCCCAGCCGGGGTTCCCGCTGGCGCAATTGAAAACTTTCGTCGATCAGGAATTT
GCCCAAATAAACATGTCCTGCATGGCATTAGTTTGTTGGGGCAGTGCCCGGATAGCATCAACGCTGCGC
TGATTTGCCGTGGCGAGAAAATGTCGATCGCCATTATGGCCGGCGTATTAGAAGCGCGCGGTCACAACGT
TACTGTTATCGATCCGGTCGAAAAACTGCTGGCAGTGGGGCATTACCTCGAATCTACCGTCGATATTGCT
GAGTCCACCCGCCGTATTGCGGCAAGCCGCATTCCGGCTGATCACATGGTGCTGATGGCAGGTTTCACCG
CCGGTAATGAAAAAGGCGAACTGGTGGTGCTTGGACGCAACGGTTCCGACTACTCTGCTGCGGTGCTGGC
TGCCTGTTTACGCGCCGATTGTTGCGAGATTTGGACGGACGTTGACGGGGTCTATACCTGCGACCCGCGT
CAGGTGCCCGATGCGAGGTTGTTGAAGTCGATGTCCTACCAGGAAGCGATGGAGCTTTCCTACTTCGGCG
CTAAAGTTCTTCACCCCCGCACCATTACCCCCATCGCCCAGTTCCAGATCCCTTGCCTGATTAAAAATAC
CGGAAATCCTCAAGCACCAGGTACGCTCATTGGTGCCAGCCGTGATGAAGACGAATTACCGGTCAAGGGC
ATTTCCAATCTGAATAACATGGCAATGTTCAGCGTTTCTGGTCCGGGGATGAAAGGGATGGTCGGCATGG
CGGCGCGCGTCTTTGCAGCGATGTCACGCGCCCGTATTTCCGTGGTGCTGATTACGCAATCATCTTCCGA
ATACAGCATCAGTTTCTGCGTTCCACAAAGCGACTGTGTGCGAGCTGAACGGGCAATGCAGGAAGAGTTC
TACCTGGAACTGAAAGAAGGCTTACTGGAGCCGCTGGCAGTGACGGAACGGCTGGCCATTATCTCGGTGG
TAGGTGATGGTATGCGCACCTTGCGTGGGATCTCGGCGAAATTCTTTGCCGCACTGGCCCGCGCCAATAT
CAACATTGTCGCCATTGCTCAGGGATCTTCTGAACGCTCAATCTCTGTCGTGGTAAATAACGATGATGCG
ACCACTGGCGTGCGCGTTACTCATCAGATGCTGTTCAATACCGATCAGGTTATCGAAGTGTTTGTGATTG
GCGTCGGTGGCGTTGGCGGTGCGCTGCTGGAGCAACTGAAGCGTCAGCAAAGCTGGCTGAAGAATAAACA
TATCGACTTACGTGTCTGCGGTGTTGCCAACTCGAAGGCTCTGCTCACCAATGTACATGGCCTTAATCTG
GAAAACTGGCAGGAAGAACTGGCGCAAGCCAAAGAGCCGTTTAATCTCGGGCGCTTAATTCGCCTCGTGA
AAGAATATCATCTGCTGAACCCGGTCATTGTTGACTGCACTTCCAGCCAGGCAGTGGCGGATCAATATGC
CGACTTCCTGCGCGAAGGTTTCCACGTTGTCACGCCGAACAAAAAGGCCAACACCTCGTCGATGGATTAC
TACCATCAGTTGCGTTATGCGGCGGAAAAATCGCGGCGTAAATTCCTCTATGACACCAACGTTGGGGCTG
GATTACCGGTTATTGAGAACCTGCAAAATCTGCTCAATGCAGGTGATGAATTGATGAAGTTCTCCGGCAT
TCTTTCTGGTTCGCTTTCTTATATCTTCGGCAAGTTAGACGAAGGCATGAGTTTCTCCGAGGCGACCACG
```

4.2 実験

実装した並列処理での処理速度向上率の確認を目的とする実験を行った．実験内容を以下に示す．

- ノード数を 1, 2, 4, 8 台の場合に分け，各ノード数において　並列化したマッピングツールを動作させ，処理時間の計測を行う．
- 各ノード数においてそれぞれ 50 回計測を行い，平均値をそのノード数における処理時間とする．

4.3 評価

処理時間の向上の評価のために並列化による速度向上の理論値と比較を行った．

並列化による速度向上比の理論値は，以下に示すアムダールの法則[6]により求められる．

$$X = \frac{1}{(1-P) + \frac{P}{N}}$$

（X：理論値, P：プログラム全体の並列化率, N：ノード数）

P = 0.7 での各ノード数における速度向上比の理論値を図 2 に示す．ノード数の増加に伴い速度比は向上していくが徐々に向上率が鈍くなり，3.33 倍程度が限界である．

この理論値と実験で得られた処理時間を比較し，並列化による台数効果の評価を行った．

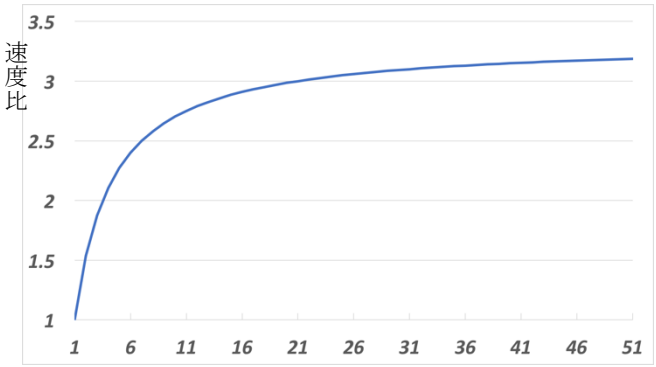

図 2　並列化率 0.7 での各ノード数における速度向上比の理論値

5. 結果

各ノード数での実行時間の計測結果と速度向上率を表 2 に，理論値との比較の結果を図 3 に示す．

表 2 各ノード数における実行時間と速度向上比，および理論値

ノード数	1	2	4	8
実行時間	1141.69	888.36	704.28	553.62
速度向上比	1.00	1.29	1.63	2.01
理論値	1.00	1.54	2.11	2.58

図 3 は各ノード数における実行時間，ノード数 1 台の時を 1.0 とした時の速度向上比，また，各ノード数における並列化後の速度向上比の理論値を示したものである．最大 8 台の並列化により処理速度は 2.01 倍となり，ノード数の増加に応じて処理速度の向上が確認できた．ただし，理論値との間には差が見られた．

6. 考察

今回得られた結果に関して考察する．マッピング処理の

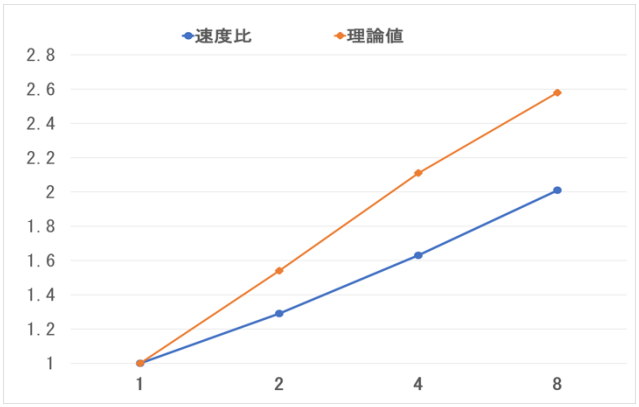

図 2　各ノード数における実行時間と速度向上比，および理論値

並列化により，ノード数の増加に伴う処理速度の向上が確認できた．最大ノード数 8 台での速度向上比が 2.01 倍となり，理論値からは離れているが，台数の増加に伴い速度向上比が比例して上昇していることから，台数効果を得られていると考えられる．速度向上比が理論値よりも小さいのは使用した接続回線が 100Mbps Ethernet であり，回線速度が遅いことがボトルネックとなっていると考えられる．したがって，より回線速度の速い 1,000Mbps Ethernet(ギガビットイーサネット)や，Serial ATA を接続に用いること でより理論値に近い速度が期待できると考えられる．

アムダールの法則からプログラム全体の並列化率 0.7 のとき，台数を無限大に増やしても 3.33 倍程度 が限界である．仮に，並列化率が 0.9 で 10 倍，0.99 で 100 倍，0.999 の場合であっても最大 1000 倍が限界である．プログラムのほとんどを並列化しても最大 1000 並列の性能が限界である．さらに，並列処理ではノード数に比例して実行時間が減っていくわけではなく，処理の中には並列に処理することにより高速化できる部分と，そうでない部分が存在するため，一定のラインを超えると並列処理により高速化できない部分がネックとなり，処理時間の減少の伸びが悪く減少の伸びは期待できない．そのため，並列計算機の実装には費用対効果の評価が重要である．

今回使用した Raspberry Pi は 1 台あたり$ 35 と安価であり，複数台の入手が容易である．また，Raspberry Pi の CPU アーキテクチャである ARM の CPU は性能に対して消費電力が低いという特徴があるため，ランニングコストを低く抑えることができる．ただし，1 台あたりの性能はそれほど高くないため，並列計算機の導入と並列プログラミングの学習用に用いるのが適していると考えられる．高性能なマシンの製作のためにマシン 1 台あたりの性能を高いものにすると金銭的コストが大きくなる．低コストで高性能なクラスタリングマシンの導入の方法として，使用目的を明確にし，専用マシンを作成することでコストを抑えて高性能なマシンを作成することが可能である

7. おわりに

本稿では PC クラスタ上で並列化したマッピングツールを動作させ，処理速度の違いを検証した．結果から MPI による並列化により処理速度の向上が確認できた．今後の課

題として，プログラム全体の並列化率の向上と，複数台のマシン間のプロセス並列だけでなくCPU内でのスレッド並列を組み合わせたハイブリッド並列を用いた場合の速度向上比の検証，マッピング処理に特化させることで性能の向上を図る等があげられる．また，ハードウェアに依存した速度向上だけではなく，アルゴリズムの改善による処理時間の短縮を図る必要がある．

参考文献

[1] Burrows M, Wheeler DJ. Technical report 124. Palo Alto, CA: Digital Equipment Corporation; 1994. A blocksorting lossless data compression algorithm.

[2] Li H and Durbin R. Fast and accurate short read alignment with Burrows-Wheeler trans- form. Bioinformatics, vol.25, no.14, pp.1754-1760. Jul. 2009.

[3] B. Langmead, C.Trapnell, M.Pop, and S.L.Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology, vol.10, no.3, p.10:R25. 2009.

[4] National Center for Biotechnology Information(NCBI). https://www.ncbi.nlm.nih.gov

[5] Richard L, Graham. Open MPI: A High-performance, Heterogeneous MPI

[6] Amdahl, Gene. Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities AFIPS Conference Proceedings (30): 483-485.