Vol.2011-BIO-26 No.2
2011/9/13

GPU を用いた配列相同性検索ツールのマルチ GPU 向け最適化

坂田幸佑[†1]　鈴木脩司[†2]
石田貴士[†2]　秋山泰[†1,†2]

感度の高い配列相同性検索を必要とするメタゲノム解析では、大量の DNA 断片配列を短時間で出力する次世代シークエンサーのデータの解析に非常に多くの計算時間を要する。この問題に対して、我々は GPU を用いることで高速に配列相同性検索を実行可能な GHOSTM システムを開発したが、依然として現実時間での解析は困難であった。そこで本研究では、多数の GPU を利用することで高速化を試みた。各計算ノードに複数の GPU が搭載されたシステムを想定し、まず GHOSTM の 1 ノード内での使用メモリを考慮した並列化実装を行い、次に並列化実装したものを、さらに複数ノードで自動処理するシステムを開発した。その結果、24 枚の GPU を使用する事で次世代シークエンサーが 1 日に出力するデータを約 10 時間程度で解析が可能となった。

Optimizing GPU based homology search tool for multi-GPU environment

Kousuke Sakata,[†1] Shuji Suzuki,[†2] Takashi Ishida[†2]
and Yutaka Akiyama[†1,†2]

Large amount of homology searches are required for analyzing vast fragment sequences produced by a next-generation sequencer in metagenomics. Thus, we developed fast GPU based homology search tool (GHOSTM) in our previous research. However, the performance of the tool was insufficient for processing a data obtained from a next-generation sequencer in real time. Therefore, in this study, we attempted to speed-up it by using many GPUs. First, we reimplemented GHOSTM to use multiple GPUs on a single node. Then, we developed automatic system to run the reimplemented tool on a number of nodes. As results, the system with 24GPUs enabled us to analyze fragment sequences produced by a next-generation sequencer in a day within about 10 hours.

1. はじめに

DNA やタンパク質配列に対する配列相同性検索は、進化の過程の情報、類似した機能や構造の情報を得るために用いられるが、メタゲノム解析においては、感度の高い配列相同性検索を大量に実行する必要があり、それが解析におけるボトルネットの一つとなっている。本研究では、このボトルネックを解決するために、メタゲノム解析における配列相同性検索を高速化することを目的とする。

土壌や腸内などの環境サンプルから様々な微生物群の DNA 配列を、それぞれの微生物を分離培養せずに混合したままで決定する解析をメタゲノム解析と呼ぶ。メタゲノム解析は通常の単一生物に対するゲノム解析とは異なり、環境中に含まれる微生物全てのゲノム配列が既知であることは稀であるため、遠縁のゲノム配列しか参照できないことが多い。そのため単一生物におけるゲノム解析で行う配列マッピングに比べて、文字列間の不一致やギャップ（挿入、欠失）をより多く許容する必要があり、さらに検索の感度を増すために、DNA 配列をタンパク質配列に変換してから解析が行われる[1)]。このためメタゲノム解析には配列相同性検索が必要となり、それには非常に多くの計算時間を要する。

近年、次世代シークエンサーと呼ばれる高スループットな DNA 配列読取装置が登場し、1 日あたり数十億塩基もの大量の配列データを解読可能となった[2)]。このため、メタゲノム解析では次世代シークエンサーから得られる大量のメタゲノムデータに対して、上記のような感度の高い解析を実施しようとすると、現実時間で行うことが困難となっている。例えば、従来の研究では配列相同性検索には一般的な配列相同性検索で用いられてきた BLAST[3)] が用いられていたが[4)]、BLAST が比較的高速に配列比較が可能な近似的手法であるにも関わらず、次世代シークエンサーが 1 日に出力するメタゲノムデータの解析には約 500 日/CPU を要していた。

この問題に対して、鈴木らは高速に大量の配列を解析可能な GPU（Graphics Processing Unit）を用いた配列相同性検索ツールである GHOSTM[5)] を開発した。GHOSTM は CPU では非常に多くの計算時間を要する配列相同性検索（特にアラインメント処理）を GPU に

†1 東京工業大学 工学部情報工学科
Depart of Computer Science,Tokyo Institute of Technology
†2 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

担当させることで、BLAST に比べて十分な検索感度を保ちながら、約 50 倍の高速化を達成した。しかしそれでも、次世代シークエンサーが 1 日に出力するメタゲノムデータの解析には約 10 日を要し、依然として現実時間での解析が困難であった。
一方、東京工業大学が保有するスーパーコンピュータ TSUBAME2.0 には大量の GPU が搭載されているため、多くの GPU を同時に利用すれば高い演算性能を得ることが可能となる。
そこで本研究では、TSUBAME2.0 上で GHOSTM を複数の GPU を用いて実行することで計算時間の削減を試みた。そのために、まず複数の GPU が搭載された各ノード上で、GHOSTM が複数の GPU を利用可能となるよう再実装を行い、その後、その再実装されたツールをさらに複数のノードで実行するシステムを開発した。

2. GPU を用いた配列相同性検索ツール（GHOSTM）について

GHOSTM は CUDA（Compute Unified Device Architecture）を用いて実装されたプログラムであり、CPU で非常に多くの計算時間を要する処理を GPU に処理させることで計算時間を削減している。CUDA は、NVIDIA 社が提供する GPGPU（General Purpose Graphics Processing Unit）を目的とした C 言語向けの統合開発環境である。
GHOSTM では、一般的な配列相同性検索で用いられてきた NCBI BLAST に含まれている BLASTX プログラムと同様に、DNA 配列をタンパク質配列に変換してから、タンパク質配列データベースに検索を行う。
GHOSTM の基本的なアルゴリズムについて述べる。まず、データベース（DB）配列から 1 文字ずらしで K-mer（長さ K の文字列）に区切り作成された key と、クエリ配列から s 文字ずらしで K-mer に区切り作成された key を用いて、それらの完全一致する場所を配列相同性検索の候補として探す。候補の探索では、クエリと DB 間で同じ key が近接して出現する個数が、ある閾値 t 以上となった領域を報告する。次に、その候補周辺で動的計画法（Smith-Waterman アルゴリズム[6]）によって精密にアラインメントを行い、そのスコアを計算する。そして最終的に結果として、クエリ毎にスコアの高い上位のものを出力する。また、GHOSTM では一度に全てのクエリと DB に対して配列相同性検索を行うのではなく、GPU のメモリを考慮して問題を分割して計算を行う。つまり、DB とクエリについて、それぞれをある一定の容量で分割してチャンクを生成し、各チャンクについて配列相同性検索を行うことで GPU メモリのオーバーフローを防止する。

3. GHOSTM のノード内並列化実装

本研究では、シングル GPU 向けに開発された GHOSTM（以下シングル版）を、まず 1 ノード内に複数の GPU が搭載されている環境で、複数の GPU を利用しての実行が可能となるよう再実装を行った。以下にその詳細を説明する。

3.1 ノード内並列

大量の配列に対する配列相同性検索では、各クエリの配列相同性検索が独立しているため、クエリを分割して処理することで、データ並列による処理が可能となっている。ノード内に複数の GPU が搭載されている環境で、各 GPU へアクセスするためには、GPU の個数と同数の CPU プロセスまたは CPU スレッドを経由する必要がある。本研究ではスレッド間でホストメモリを共有できるという性質を考慮してスレッドによる実装を選択し、POSIX 標準である POSIX threads（Pthreads）ライブラリを用いた。これにより、プロセス内のメインスレッドから各 GPU にアクセスするスレッドと、スレッド間の処理を制御する更新用スレッドを生成する。

3.1.1 スレッド間でのメモリ共有

プロセス内に複数のスレッドを生成するマルチスレッドプログラミングでは、プロセス中の全スレッドは同じメモリ空間に存在している。そこで本研究では、各クエリの参照する DB は共通であることから、各スレッド間で DB チャンクを共有することで、ホストメモリの使用量を軽減している。**図 1** に示すように、各スレッドが分割されたクエリチャンクを担当し、DB チャンクは各スレッド間で共有されている。

3.1.2 同 期 処 理

並列処理を正しく行うためには、DB チャンクの切り替え（**図 2**）と各スレッドの計算結果のマージ処理（**図 3**）において同期処理を行う必要がある。DB チャンクの切り替えでは、各スレッドは DB チャンクに対してクエリチャンクの配列相同性検索を行うため、スレッド間で同期が行われない場合は計算結果に矛盾が生じる。計算結果のマージ処理では、クエリチャンク毎に出力される計算結果を、各スレッド間で同期することで、シングル版と等しい計算結果を得ることが可能となる。

4. 複数ノードを用いた大規模解析システムの実装

TSUBAME2.0 では複数のノードを利用可能であるため、単一ノード内で複数の GPU を利用しての実行が可能となった GHOSTM を、さらに複数のノードで並列実行すること

Vol.2011-BIO-26 No.2
2011/9/13

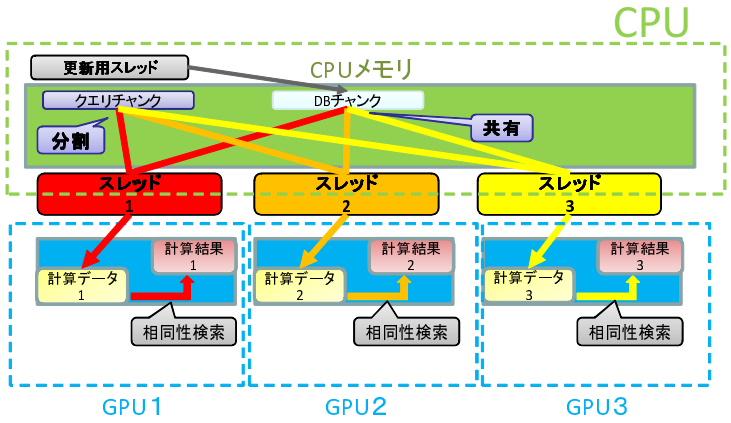

図 1 DB チャンクの共有
Fig. 1 Share of DB chunk

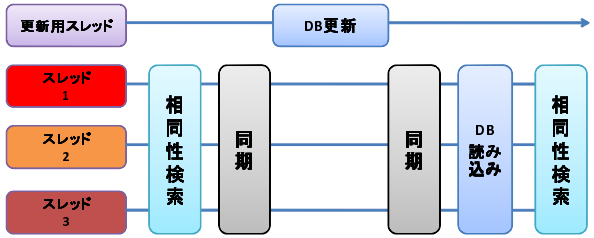

図 2 DB チャンク更新における同期処理
Fig. 2 Synchronization on Update of DB chunk

で、解析時間を更に削減することが可能となる。本研究では、複数ノードの利用は PBS (Portable Batch System)[7] を利用したバッチ処理により、各ノードにジョブを展開させることで実現した。ここで本稿では、並列化された GHOSTM を各ノードに展開するまでの一連の処理を自動化したシステムを “大規模解析システム” と呼ぶことにする。多数の GPU を利用することが可能である大規模解析システムは、次世代シークエンサーから得られる大量のメタゲノムデータを現実時間で解析可能である。

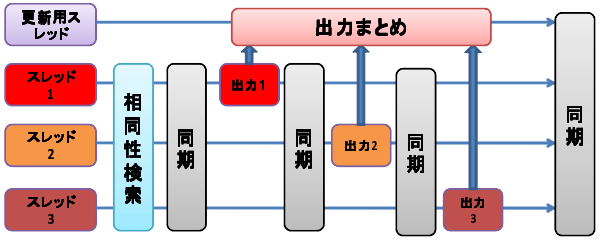

図 3 出力結果のマージにおける同期処理
Fig. 3 Synchronization on merging output results

4.1 ノード間並列

本研究で採用したノード間並列の手法は、PBS を利用したバッチ処理により、複数ノード上でジョブを展開させることで並列処理を実現した。各ノードでは、並列化された GHOSTM が分割されたクエリファイルを担当するため、PBS を利用したバッチ処理にアレイジョブの機能を用いた。アレイジョブは各ノードに展開されるジョブに個々にパラメータを与えることが可能な機能であり、本研究のように各ノードで異なるデータを計算させたい場合に適しているためである。

本研究で複数ノードにまたがる並列処理に、MPI ライブラリではなく、バッチ処理を選択したが、その理由として以下のようなことが挙げられる。第 1 に各ノードに展開される並列化された GHOSTM の計算が、ノードレベルで独立しているという点である。これによりメッセージパッシングを使用したノード間での通信を行う必要がないため、各処理が独立しているバッチ処理を用いることができる。第 2 に複数ノードの使用が容易な点である。MPI では指定したノード数を占有して並列処理を行うため、8 ノードを指定すると、8 ノード全てが使用可能となるまで実行を開始できない。しかし、バッチ処理では使用できるノードに逐次ジョブを投入して計算を開始可能なため、8 ノードが同時に全て使用可能でない場合も、計算が終了したノードを逐次使用することで 8 ノード分の計算を行うことが可能となる。以上の理由により本研究ではバッチ処理を選択した。

4.2 大規模解析システム

並列化された GHOSTM の複数ノードでの実行を実現するには、複数のステップを踏む必要がある。第 1 にクエリファイルの分割を行う。各ノードではこの分割クエリファイルを並列版を用いて配列相同性検索を行う。第 2 に分割されたクエリファイルに対する前処理

が必要である。これにより各分割クエリファイルに対するクエリチャンクを生成する。第 3 に各クエリデータと DB を用いて、各ノードにジョブとして展開する。以上で各ノードで配列相同性検索を行うことが初めて可能となる。しかし、これらのステップを人間が逐次行うことは非常に煩雑である。そこで、これらの処理を python スクリプトによって自動化したシステムを開発した。また、並列版に必要なパラメータは非常に多いので、これらを設定ファイルにまとめ、一括の設定を可能とした。以下に大規模解析システムの大まかな実行フローを説明する。

(1) クエリファイルの分割
クエリファイルのデータは FASTA 形式であり、ヘッダ行とシーケンス文字列の対応関係を保ちつつ、使用ノード数だけ配列単位で分割を行う。

(2) 分割クエリファイルの前処理
分割された各クエリファイルに前処理を行い、それぞれのクエリチャンクを生成する。

(3) 複数ノードにおける配列相同性検索
分割クエリファイルより生成された各クエリチャンクと DB 配列に前処理を行って生成した DB チャンクを、アレイジョブを用いて各ノードに展開する。これにより各ノードは担当するクエリの配列相同性検索を行う.

(4) 結果のマージ
大規模解析システムの各ノードにおける出力は分割されたクエリに対するもので、クエリの分割を行わないシングル版の出力と等しくするためには、各ノードから得られた出力を UNIX コマンドの cat を用いて連結を行う。

5. 実験結果

5.1 計算機環境と使用データ

5.1.1 計算機環境

実装と実験には TSUBAME2.0 を用いた。TSUBAME2.0 は東京工業大学 GSIC が保有しているスーパーコンピュータであり、2011 年 6 月に発表されたスーパーコンピュータの性能ランキングにおいては世界 5 位を記録している[8]。TSUBAME2.0 を構成している各ノードには、NVIDIA 社の GPGPU 専用の GPU である Tesla M2050 が 3 枚搭載されており、さらにこれらを 1408 ノード結合することで、理論最大性能としては 2.4PFLPOS の性能を実現している。また、TSUBAMA2.0 の特徴は CPU と GPU の演算性能の比であり、CPU が 0.2PFLOPS であるのに対して GPU が 2.2PFLOPS ということで、GPU の比重が非常に大きい。このため TSUBAME2.0 に搭載されている GPU を大量に用いることで、高い演算性能を得ることが可能となる。**表 1** は TSUBAME2.0 の 1 ノードの構成をまとめたものである。コンパイルに用いた CUDA のバージョンは 3.2 であり、gcc のバージョンは 4.3.4 である。

CPU	Xeon X5670（2.93GHz）	12 コア
memory	54GB（一部 96GB）	
GPU	NVIDIA Tesla M2050（1.15GHz）	3 枚

表 1 TSUBAME2.0 の 1 ノード（Thin node）における環境説明
Table 1 Performance of a single node (Thin node) on TSUBAME2.0

5.1.2 使用データ

本研究で用いた配列相同性検索の検索対象となるデータベースは、京都大学 KEGG[9] からダウンロードした “gene.pep” というタンパク質配列であり、配列の本数は約 460 万本、全配列の合計長は約 15 億残基である。また、クエリファイルは東工大大学院生命理工学研究科の黒川顕教授のグループから頂いた次世代シークエンサーによって得られた 60 塩基対の 1000 万本の土壌のメタゲノム DNA 断片配列である。各実験では、その中から比較用に、1000、1 万、10 万、100 万、1000 万本をランダムに選択し使用した。

5.1.3 GHOSTM の実行時パラメータ

シングル版に適用したパラメータは、先行研究[5] で配列相同性検索の感度と速度が十分であると示された値であり、以下にそれを示す。クエリチャンクのサイズは 64MB、DB チャンクのサイズは 128MB、インデックスとクエリに用いる K-mer のサイズは 4、クエリを K-mer で区切る際にずらす文字数 s は 2 である。配列相同性検索の際に、いくつ key があれば候補にするかを指定する閾値 t の値は 2、DB を区切る領域の大きさ r の値は 4 である。アラインメントスコアを計算する際のスコア行列は PAM30[10]、ギャップペナルティは −8 を用いた。

5.2 1 ノード上での比較

それぞれのクエリサイズに対して、1GPU のみを使用する GHOSTM（以下シングル版）と GPU 数を 3 つ使用する並列化された GHOSTM（並列版）について、計算時間、メモリ使用量について各 3 回の計測を行った。

5.2.1 計算時間

並列化による速度向上を確認するために、シングル版と並列化された GHOSTM につい

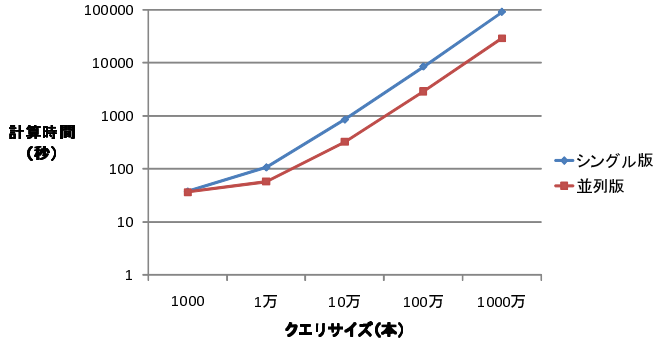

図 4 シングル版と並列版 GHOSTM（GPU 数=3）の平均計算時間（秒）
Fig. 4 Average computation time of single and parallel version of GHOSTM(♯GPUs=3) (sec.)

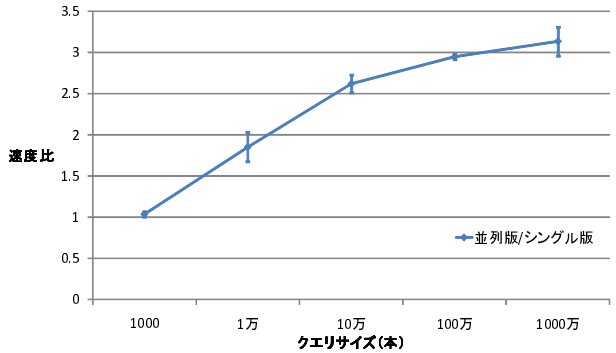

図 5 シングル版に対する並列版 GHOSTM（GPU 数=3）の速度比
Fig. 5 Speed-up ratio of parallel version of GHOSTM(♯GPUs=3) compared to single version

て実行時間を計測した。実行時間の計測には、PBS から出力されるジョブ実行情報に記述された計算時間を表す “walltime” の値を使用した。**図 4** は平均計算時間を、**図 5** は並列版/シングル版の速度比を表している。

5.2.2 メモリ使用量

並列化された GHOSTM で DB をスレッド間で共有した事によるメモリ使用量の軽減を

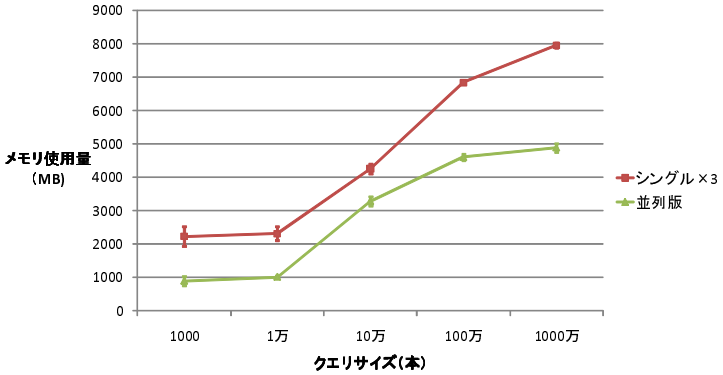

図 6 シングル× 3 と並列版 GHOSTM（GPU 数=3）の平均メモリ使用量（MB）
Fig. 6 Average memory usage of single × 3 and parallel version of GHOSTM(♯GPUs=3) (MB)

確認するために、実行時のメモリ使用量を計測した。ここで「シングル× 3」とは、シングル版を 3 つの別のプロセスとして実行させた場合の、3 プロセスの実行時のメモリ使用量の合計である。合計で各クエリサイズを計算するものである。メモリ使用量の計測には、PBS から出力されるジョブ実行情報に記述された使用メモリを表す “used memory” の値を使用した。**図 6** は平均メモリ使用量を表している。

5.3 複数ノードを用いた大規模解析システム

複数ノードを使用した場合の速度向上を確認するために、大規模解析システムを用いて実行時間を計測した。また、ノード間での並列処理の効果を確認するために、図 5 よりシングル版に対する速度比が約 3 となった 100 万,1000 万本を使用し、ノード数は 1,2,4,8 で実験を行った。実行時間の計測には、time コマンドを用いて大規模解析システムの実行開始から終了までの時間を計測し、各 3 回の計測を行った。**表 2** は平均計算時間を、**図 7** は複数ノード/1 ノードの速度比を表している。

	1 ノード	2 ノード	4 ノード	8 ノード
100 万本	2882 （約 48 分）	1469 （約 24 分）	766 （約 12 分）	410 （約 7 分）
1000 万本	28999 （約 8 時間）	14737 （約 4 時間）	7436 （約 2 時間）	3847 （約 1 時間）

表 2 大規模解析システムを用いた際の平均計算時間（秒）
Table 2 Average computation time with parallel version of GHOSTM(♯GPUs=3) on a number of nodes (sec.)

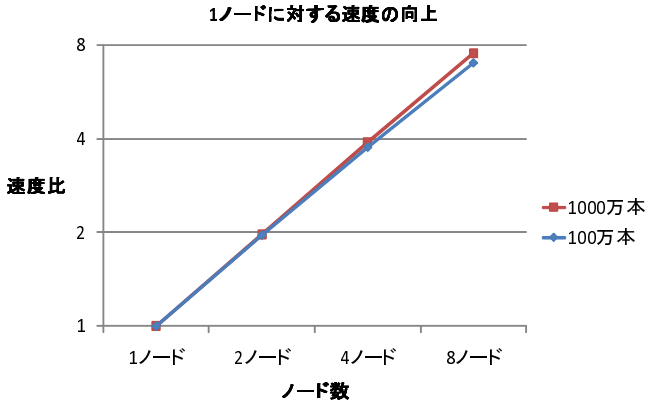

図 7 複数のノードによる速度の向上
Fig. 7 Speed-up ratio of parallel version of GHOSTM(♯GPUs=3) on a number of nodes compared to a single node

6. 考　　察

6.1 1 ノード上での比較

6.1.1 計 算 時 間

図 5 に示すように、シングル版に対する並列化された GHOSTM の速度比は、クエリが 1 万本以下の場合、GPU を 3 つ使用しているにも関わらず 3 に達していない。これについては、並列化された GHOSTM の実行時の GPU の初期化や I/O が多くの時間を要しているためと考えたが、実際にそれらを計測したところ数秒程度であったため、GPU に大量に搭載されているコアを有効に占有できていない（占有率が低い）のではないかと推測している。クエリの本数が 10 万本以上の場合については、10 万、100 万、1000 万本の各データが理論値である約 3 倍の速度比に達し、クエリサイズと計算時間の関係に線形な関係を確認できる。これはクエリサイズが大量なため、GPU に大量に搭載されているコアを十分に占有できている（占有率が高い）ためであると考えられる。

6.1.2 メモリ使用量

図 6 に示すように、並列化された GHOSTM のメモリ使用量は、クエリサイズが 10 万本以下の場合、シングル版を 3 つの別のプロセスで並列実行した場合に対して、DB データ分のメモリ使用量の軽減を確認できる。クエリサイズが 100 万本以上の場合については、並列化された GHOSTM がシングル版を 3 つの別のプロセスで並列実行した場合に対して、DB データ分以上のメモリ使用量が軽減されている。これは実装中の様々な最適化によるものと考えられる。

6.2 複数ノードを用いた大規模解析システム

図 7 に示すように、大規模解析システムを用いた際の処理時間は、クエリサイズが 100 万、1000 万本の両者において、ノード数に対してほぼ線形な関係を確認できる。これはノード間での通信が存在しないためである。

次世代シークエンサーである Applied Biosystem 社の SOLiD4[11] が 1 日に出力するデータは約 6G（60 億）塩基となっており、これを本実験のデータに換算すると、60 塩基対の約 1 億本の DNA 配列に対応する。計算時間はクエリサイズが 10 万本以上であればクエリサイズの増加に対して線形に増加し、ノード数の増加に対して線形に減少することを考慮すれば、表 2 の結果より 1 億本の計算時間は 8 ノード（24GPU）を使用した場合、約 10 時間程度で解析が終了すると推定できる。

Vol.2011-BIO-26 No.2
2011/9/13

7. 結　　論

本研究では、先行研究で開発されたGPUを用いた配列相同性検索ツールであるGHOSTMを複数のGPUを用いて大規模に実行するシステムを提案し、TSUBAME2.0上でこの実装を行った。1ノード内で複数のGPUを利用可能となったGHOSTMでは、GPU数が3枚の場合に、クエリサイズが10万本以上で約3倍の速度向上を確認した。メモリ使用量については、DBをスレッド間で共有する事でメモリ使用量の軽減を行った。またTSUBAME2.0上の複数ノードでの並列実行を自動化する大規模解析システムを開発した。これによりTSUBAME2.0が搭載している多数のノードを用いた大規模解析が可能となり、次世代シークエンサーが1日に出力するデータの解析が、8ノード（24枚のGPU）を使用した場合に約10時間程度で解析が可能となった。

7.1 今後の課題

TSUBAME2.0には多数のノードが存在するため、今回の実験よりも更に使用ノード数を増加させることで計算時間の削減が可能である。しかし、大規模解析システムを用いて、多数のノードでプログラムを実行する場合には、ハードウェアの異常等の様々なエラーが発生することが考えられる。このため、このような場合を想定したフォールトトレラントなシステムを構築する必要がある。

謝　　辞

本研究において、貴重なデータを使わせて頂いた東京工業大学大学院生命理工学研究科生命情報専攻の黒川顕教授に御礼申し上げます。

参 考 文 献

1) Susannah, G.T. and Edward, M.R.: Metagenomics: DNA sequencing of environmental samples, *Nature Reviews Genetic*, Vol.6, pp.805-814 (2005).
2) Wooley, J.C., Godzik, A. and Friedberg, I.: A Primer on Metagenomics, *PLoS Computational Biology*, Vol.6, No.2 (2010).
3) Altschul, S.F., Gish, W., Miller, W., *et al.*: Basic local alignment search tool, *Journal of Molecular Biology*, Vol.215, pp.403-410 (1990).
4) Diana, L.C., Sean, C., Edward, C.H., *et al.*: A Metagenomic Survey of Microbes in Honey Bee Colony Collapse Disorder, *Science*, Vol.318, pp.283-287 (2007).
5) 鈴木 脩司, 石田 貴士, 秋山　泰: GPUによるDNA断片配列の高速マッピング, 情処研報, 2010-BIO-21, Vol.30, pp.1-6 (2010).
6) Smith, T.F. and Waterman, M.S.: Identification of Common Molecular Subsequence, *Journal of Molecular Biology*, Vol.147, pp.195-197 (1981).
7) Robert, L.H.: Job Scheduling under the Portable Batch System, *Lecture Notes in Computer Science*, Vol.949, pp.279-294 (1995).
8) TOP 500 SUPERCOMPUTER SITES: `http://www.top500.org/list/2011/06/100`
9) Kanehisa, M., Goto, S., Furumichi, M., *et al.*: KEGG for representation and analysis of molecular networks involving diseases and drugs, *Nucleic Acids Res.*, Vol.38, pp.355-360 (2010).
10) Dayhoff, M. and Schwartz, R.: Matrices for detecting distant relationship, Atlas of Protein Sequences, pp.353-358 (1979).
11) Applied Biosystems SOLiD4 System: `http://www3.appliedbiosystems.com/cms/groups/global_marketing_group/documents/generaldocuments/cms_078637.pdf`