Vol.2019-HPC-172 No.2
2019/12/18

分散並列相同性検索システムGHOSTZ PW/GFの大規模環境における評価

町田 健太[1,a)]　建部 修見[2]

概要：ゲノムデータは，代表的なビッグデータの一つであり，次世代シーケンサの性能向上や普及により得られるデータの量は年々増大している．ゲノム解析の一つである相同性検索は，未知の DNA 配列をクエリとして，既知の DNA 配列データベースから相同なものを収集する手法であるが，ゲノム解析の過程において従来の手法ではその処理速度がネックになってしまっている．本研究では，分散ファイルシステムである Gfarm と動的なワークフローエンジンである Pwrake を基盤とした分散並列相同性検索システムを提案・実装し，TSUBAME3.0 において評価を行った．結果として，提案システムは高いスケーラビリティを示し，180 ノードを用いた実験ではビルド済みの約 1 億レコードで構成される non redudant DB と約 5 億レコードから構成されるシーケンスデータに対しておよそ 2 時間で相同性検索を実行した．

1. はじめに

細菌は地球上の生物を含むあらゆる環境中に存在しており，周囲の環境と共生・総合扶助の関係を持ち，独自のエコシステムを築き上げている．ヒトは自身の細胞数の 10 倍以上に相当する細菌を有しているとされており，それらの細菌叢はヒトが食べた物を栄養分に代謝したり，病原性細菌からの感染を防いだりと，ヒトの健康状態の維持や発病の抑制に関与している [1]．一方，これらの細菌叢の異常が人体に及ぼす影響も大きく，生活習慣病やがん，アレルギー，自閉症など様々な疾患の原因となることが明らかになっている．そのため，細菌叢の群集構造や機能を明らかにすることがこれらの疾患の治療や予防に繋がると期待されているが，地球上のほとんどの細菌は培養が困難であるため，従来の手法では解析が容易ではなかった．

メタゲノム解析は，ある環境中から直接得られたゲノム DNA を用いて，その集合構造を明らかにすることにより，遺伝子プールの変動や環境との相互作用の解明を可能にしたゲノム解析手法である [2]．メタゲノム解析は細菌に対する培養過程を経ずに細菌叢から直接そのゲノム DNA を調製する事が可能であるため，ヒトの体内の微生物群集構造を明らかにする手段として有用である．また，メタゲノム解析は数千種類もの生物を同時に解析することが可能な次世代シーケンサー (NGS) テクノロジーと親和性が高く，近年その関連論文数は増加し続けており [3]，メタゲノム関連市場のさらなる活性化も予測されている [4]．

メタゲノム解析のプロセスは大まかに以下のステップを踏む．まず初めに，被験者のメタデータとしてサンプルを取得し，そのサンプル中から細菌叢 DNA を NGS により直接シーケンシングする．さらに，シーケンシング結果のリードをクオリティなどでフィルタリングする．続いて，KEGG[5][6] や COG[7] 等が提供するリファレンス DB に対し類似度や相同性の検索を行うことで，各遺伝子の機能や代謝経路等の情報を収集し，含まれる種の構成と存在量などを知る．そして，必要であれば可視化やさらなる解析を実施し，環境中の微生物や細菌の群集構造や機能を推測する．

近年，NGS の性能向上や普及により，リファレンス DB だけではなくサンプルから取得されるゲノムデータの量も爆発的に増大しており，DNA の相同性の検索ステップにおいて，従来の解析手法では膨大な時間を要してしまう点が問題となっている．

2. 背景

相同性検索ツールとして，処理の高速化を目指して BLAST[8] や GHOSTX[9]，GHOSTZ[10]，DIAMOND[11] 等といった様々なアリゴリズムを用いたソフトウェアの開発が現在に至るまで行われている．しかし，これらのツールは単一ホスト上での実行を想定して開発されており，増大するゲノムデータ，特にメタゲノムデータのような大規模な入力に対して，メモリの枯渇や実行時間の肥大化が

[1] 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
[2] 筑波大学計算科学研究センター
[a)] machida@hpcs.cs.tsukuba.ac.jp

問題となってしまう．この問題に対し，相同性検索をスーパーコンピュータ等の PC クラスタ上における分散並列処理に拡張することでこれらの問題の解決を目指す研究がいくつか行われている (詳細は 4 章にて述べる)．例えば，GHOST-MP というツールは日本の理化学研究所によって運用されていた京というスーパーコンピュータや東京工業大学により運用されていた TSUBAME2.5 上での動作を想定して作成されており，実際に京において 49,152 コアを用いて実験を行っている．関連研究に見られるように，ゲノム解析におけるスーパーコンピュータの利用はもはや不可欠なものであり，各遺伝子研究機関においては独自のスーパーコンピュータを保持している場合が多い．例えば，日本国内の例を挙げると，国立遺伝学研究所では 2019 年の 3 月から可動を開始した 204 ノード (内 64 ノードが Tesla V100 を搭載し infiniband により内部接続) から構成されるスーパーコンピュータシステムを保有しており，東京大学のヒトゲノム解析センターでは，2019 年の 4 月から運用を開始した 335 ノードからなる Shirokane5 と呼ばれるスーパーコンピュータを保有している．

GHOSTZ PW/GF はスーパーコンピュータ等の大規模なクラスタ上での実行を想定した，実行時間の肥大化とメモリの枯渇の両方の問題を解決する分散並列相同性検索システムである．

このシステムは，相同性検索の際に GPU を利用しかつファイル I/O に関してクラスタの各ノードに固有の NVMe SSD 等の高速なストレージデバイスを最大限活用する設計であるため，近年のスーパーコンピュータのアーキテクチャのトレンドに追従したシステムであると言える．本研究では，このシステムを用いて TSUBAME3.0 上で大規模データに対する相同性検索の実験を行った．以降の章では，関連研究やシステムの概要，大規模環境における実験結果について述べる．

3. 提案システム

本研究では，相同性検索に用いるクエリと DB の両方をチャンクに分割し，全ての組み合わせに対して相同性検索タスクを実行する.DB はそれぞれのチャンク毎に独立して構成し，相同性検索の完了後にそれぞれの DB を用いたタスクにおける結果ファイルを集約する．このチャンクへの分割により，メモリ不足によるエラーは回避され，タスクを並列に実行することが可能になり処理が高速化される．また，増大し続けるゲノムデータに対してシステムは高いスケーラビリティが要求される．相同性検索の分散並列実行において，スケーラビリティを低下させる大きな要因は大量の通信の発生による低速な I/O であるため，I/O 性能がスケールアウトするような実行基盤が必要である．

そこで，本研究では広域分散ファイルシステムである Gfarm[19] と Gfarm と親和性の高い動的ワークフローエンジンである Pwrake[20] を用いた実行基盤を用いて，相同性検索を分散並列実行するシステムを提案する．このシステムは，Gfarm の機能によりタスクのファイルの出力先にはプロセスが実行されているノードが選択され，かつ Pwrake の動的なファイルアクセスの時間的局所性を活かすスケジューリングにより入力ファイルが存在するノードがタスク実行ノードとして選ばれるため，高速なローカル I/O を最大限活用でき I/O 性能がスケールアウトする．この実行基盤上で，強力なアルゴリズムで実装された相同性検索ツールを用いて 相同性検索を並列に実行する．

提案システムのシステム構成を**図 1** に示す．

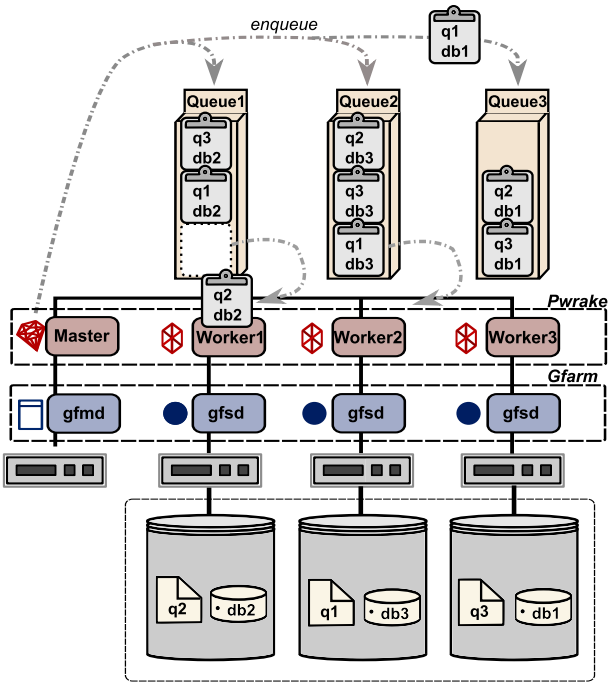

図 1 提案システムの概要

Gfarm ファイルシステムは，メタデータサーバである gfmd とファイルサーバである gfsd から構成される．gfsd によって各サーバのストレージが束ねられ，クライアントからは 1 つの名前空間として見える．実際にクライアントがファイルにアクセスする際は，メタデータサーバからファイルの所在地を聞き，ファイルサーバと直接データのやり取りを行う．

Pwrake では，マスターが Rakefile と呼ばれるタスクのワークフローが記述されたファイルを基に有向非循環グラフを生成し，ssh 接続によってワーカースレッドが動作する各ノードのタスクキューにそのノードの実行候補タスクをエンキューする．各ワーカスレッドはキューからタスクをデキューしタスクを実行する．タスクの実行スケジューリングは動的に決定されるため，タスクをできるだけ細かい粒度に分割して Rakefile を記述することで，実行時のロードバランスを高めることができる．なお，Pwrake は ruby で実装されており Rakefile は rake と呼ばれる ruby の内部

DSL を用いて記述する．

また，本研究では相同性検索ツールとして GHOSTZ-GPU を用いる．GHOSTZ-GPU は C++で実装された 2 入力 (クエリ・DB)1 出力 (相同性検索結果ファイル）の相同性検索ツールであり，OpenMP を用いたマルチスレッドによる高速化と CUDA による GPU を利用した高速化が施されている．

提案システムでは，ノードレベル/ノード内プロセスレベルの並列性は Pwrake によって管理される．ノード内の同時実行プロセス数は，プロセスの合計のメモリ使用量が RAM のサイズを超えない程度に設定する必要がある．メモリ使用量の見積もりは，GHOSTZ-GPU を用いて実験的に行った．また，スレッドレベル/GPU レベルの並列性は GHOSTZ-GPU によって管理される．ここで，本研究では GHOSTZ-GPU に対して I/O 部分の最適化のみを施し，それ以外は修正を加えずに既存のソフトウェアをそのまま用いている．なお，CPU や GPU のコアなどに対するリソース割当スケジューリングは OS に依存している．

3.1 ワークフロー

本システムにおける相同性検索のワークフローの概要を**図 2** に示す．このワークフローをタスクの依存関係として定義し，Rakefile に記述することで相同性検索を分散並列実行する．

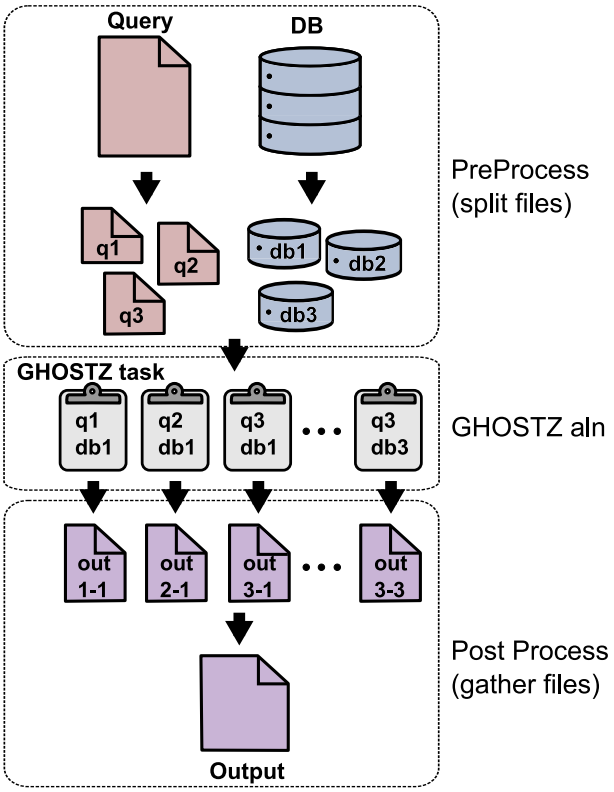

図 2 システムのワークフロー

3.1.1 前処理 (ファイル分割タスク)

システムの入力は，ゲノム配列が羅列された FASTA フォーマットのクエリファイルと DB の元となる FASTA ファイルである．初めに相同性検索の前処理としてこれらの入力に関して，ファイルを分割し各ノードに分散配置する．この際に用いるファイル分割プログラムは C++で実装し，OpenMP によって高速化を図った．分割されたファイルのうち DB の元となるファイルは GHOSTZ を用いて DB にビルドされ，全てのファイルをノード間にバランス良く配置する．またこのステップにおいて，入力ファイルの複製を複数ノードに持たせて，後の相同性検索タスクにおいて通信の発生を抑え，高速なローカル I/O による read の割合を高める．

3.1.2 相同性検索実行タスク

前処理が完了すると，GHOSTZ のアライメントを実行するタスクを並列に実行する．GHOSTZ の入力のうち DB ファイルに関しては事前にビルドされている必要があるが，ビルドは前処理タスクにおいて実行されている．このステップにおいて，ノード当たりの実行コア数やファイル複製数，タスクあたりの問題サイズ，1 タスクで用いるスレッド数等，様々なパラメータがシステム最適化の要因となる．

3.1.3 後処理 (ファイル集約タスク)

相同性検索で出力された断片的な出力ファイルを，クエリ毎に集約するタスクである．ファイルの集約タスクの依存関係は，**図 3** のような木構造になるため，並列に処理することが可能である．また出力ファイルは，2 つの入力ファイルサイズに依存して大きくなるため，必要に応じて圧縮を行う．

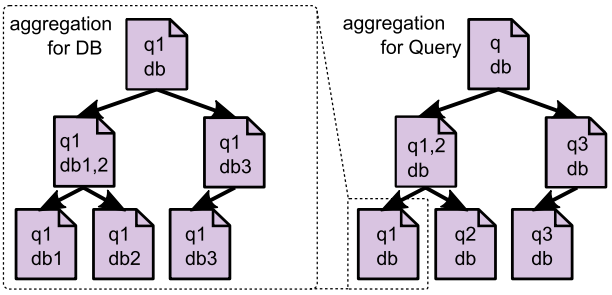

図 3 ファイル集約タスク

4. 関連研究

相同性検索をスーパコンピュータ等の分散並列実行に拡張することで，処理の高速化とメモリの枯渇という問題を解決する研究がいくつか行われている．分散処理フレームワークとして有名なものに MPI と MapReduce があり，これらを用いた研究が多い．

MPI を用いた実装では，その利点として細かな部分ま

で性能をチューニングすることができるため，比較的少量のデータセットに対する実行に関しては，一般に Hadoop や Spark における実行よりも高速に動作させることが可能である．しかしながら，MPI は通信を用いた並列処理インターフェースであるため，所謂ビッグデータと呼ばれるような大規模なデータセットに関しては，全データをメモリに載せることができず，ファイル入出力に関して別の最適化が必要となる．

Hadoop や Spark はファイルシステムとして HDFS を採用し，ファイルのローカリティを活かす設計となっているため，高速なローカルディスク I/O を生かすことが出来，通信がボトルネックとなり難いという利点がある．また，様々な高級言語がサポートされており，Hadoop では MapReduce，Spark では RDD という分散並列処理のフレームワークに落とし込んで実装するため，一般的に MPI と比較して実装のコストが低い．以下では，これらのフレームワークを用いた研究についてその概要を説明する．

4.1 GHOST-MP[14]

GHOST-MP は GHOSTX のアルゴリズムを MPI ライブラリを用いて並列化することで，京や TSUBAME3.0 などの並列分散メモリシステム上での大規模並列検索に用いられるツールである．ハイブリッド並列型の相同性検索ツールであり，ノード間の並列化には MPI を，ノード内のタスク並列化には OpenMP を用いている．ノードレベルでは master-worker モデルを採用し，クエリのシーケンスが worker プロセス数分のチャンクに分割され，master プロセスがそれぞれの worker プロセスにクエリのチャンクをタスクとして割り当てる．更にノード内ではクエリのチャンク中のシーケンスがタスクとしてキューにエンキューされていき，OpenMP のスレッドがロックを獲得してタスクをデキューすることで相同性検索の並列実行を実現している．現在公開されているものでは，各実行ノードに分散配置されたファイルを扱うのではなく，一つのクエリ・DB ファイルに対して処理を行う．

実験では，京を用いて相同性検索ツールの中で最も有名な BLAST というツールの MPI 実装である BLAST-MPI と比較を行っている．結果として，このツールは BLAST-MPI と比べて高速に動作し線形のスケーラビリティを示したが，メモリ使用量の見積もりを課題としている．

4.2 SparkBLAST[16]

この研究は，BLAST を Spark を用いて分散並列実行するというものであり，クエリはクラスタ上に分散するが DB ファイルは分散しない．Spark の pipedRDD を用いることで BLAST を実行するジョブをクラスタ上の各ノードで実行する．Google Cloud と Microsoft Azure を用いて，Hadoop ベースの相同性検索ツールである Cloud BLAST と比較を行っており，64 ノードにおいてクエリに 805[MB] のものと 11[GB] のものを用いた実験では，Cloud BLAST に対し高速に動作し，使用メモリの効率も優れていることを示している．Hadoop ベースのツールである Cloud BLAST よりも性能が優れている要因として，Spark の RDD によるインメモリ処理とそれに起因する I/O 時間の削減であると言及している．なお，クエリの分割による断片的な相同性検索の結果ファイルを集約するタスクは実装されていない．

4.3 HAMOND[17]

この研究は，DIAMOND を Hadoop を用いて分散並列処理するというものである．AWS の EMR と互換性があり，GUI での操作が可能であるため，専門知識のない生物学者向けの UI 設計がなされている．なお，DB は分散せずにクエリのみを分散する．実験では，AWS を用いて 100 ノード上でシングルホストでの DIAMOND の実行 (10 スレッド) と比較しており，HAMOND において 100 スレッドを用いた際に約 10 倍高速であることを示している．また，100 スレッドから 400 スレッド (100 ノードにおいて単一 DIAMONDO タスクあたり 4 スレッド使用) にかけて性能がスケールアウトしたことを報告している．

4.4 関連研究との比較

関連研究と比較した際の提案システムの利点の 1 つが，DB とクエリ両方を分割して，実行可能である点である．SparkBLAST や HAMOND はいずれも DB に関しては単一のものを用いており，将来的に大規模になると予想される DB に対してメモリの枯渇が問題となると考えられる．

また，提案システムは GHOSTZ-GPU を用いた実行を想定しているが，実際には，以下の実験で示すように様々なアルゴリズムで実装された相同性検索ツールを手軽に分散並列実行可能である．また，分散並列処理のロジックを実装する点において，タスクの依存関係を記述するだけであるため，MPI や Hadoop・Spark で実装されたツールと比較するとアルゴリズムの改良等による再実装のコストが小さく，拡張性が高い．

本研究では以上の 3 つのツールとの実行時間の比較を行った．比較結果を**図 4**，**図 5**，**図 6** に示す．なお，GHOST-MP との比較では DB の元ファイルに 5[GB]，クエリに 2[GB] のものを，Spark BLAST との比較では DB の元ファイルとクエリに 100[MB] のものを，HAMOND との比較ではそれぞれに 100[MB] と 1[GB] のものを用いて実験を行っている．なお，Spark においては Executor 数を 5，Hadoop においては Mapper 数を 1 にしている．また，SparkBLAST との比較では，提案システムにおいて相同性検索ツールとして BLAST を，HAMOND との比較では DIAMOND を用いるようにすることで，できるだけ同

条件で比較を行うようにしている．

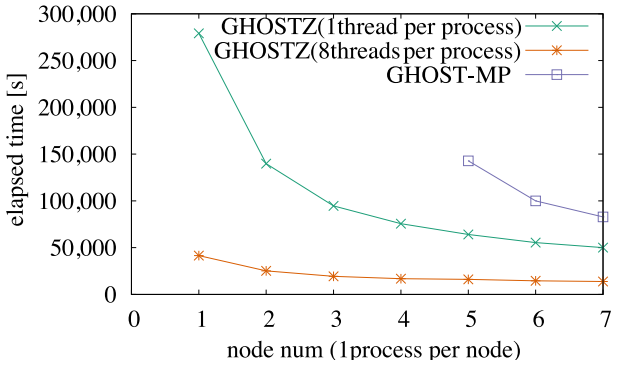

図 4 GHOST-MP との比較

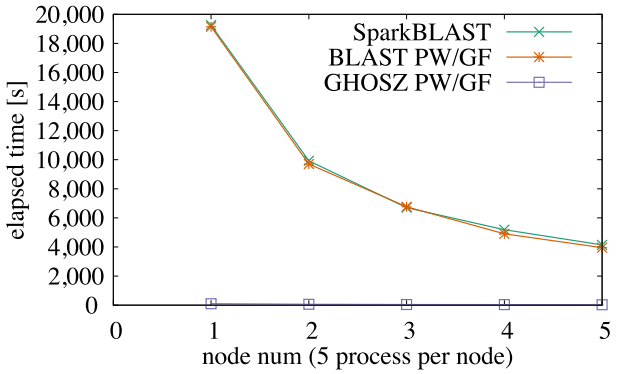

図 5 Spark BLAST との比較

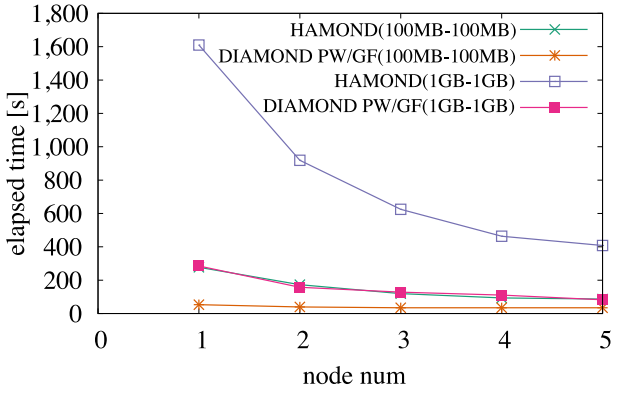

図 6 HAMOND との比較

5. 大規模実行環境における評価

本研究では，提案システムに関して東京工業大学のスーパーコンピュータである TSUBAME3.0 を用いて大規模環境における性能評価を行った．

TSUBAME3.0 の計算ノードの概要を**表 1** に示す．540 ノードで構成され各ノードは Intel Omni-Path によって内部接続されている．また，ジョブスケジューリングシステムには UNIVA Grid Engine を採用している．

提案システムは，GPU を用いて高速に動作する相同性検索ツールを用いており，またクラスタの各ノードのローカルストレージを束ねて合計の I/O 性能をスケールアウトさせる狙いがあるため，各ノードに NVMe SSD と GPU を搭載している TSUBAME3.0 における性能評価は提案システムの性能を評価するのに適している環境であると考えられる．

表 1 TSUBAME3.0 の計算ノード

OS	SUSE Linux Enterprise Server 12 SP2
CPU	Intel Xeon E5-2680 V4 Processor × 2 (28 cores)
memory	DDR4-2400 ECC REG DIMM × 8 (256GiB)
GPU	NVIDIA Tesla P100 × 4
Storage	Intel NVMe SSD (2TB)

5.1 大規模データに対する評価

5.1.1 相同性検索ステップ

大規模データにおける性能評価では，システムの入力として DB の元ファイルとして約 62[GB] のものを，クエリに 70[GB] と 250[GB] のデータを用いて実験を行った．ここで，クエリに 250[GB] のデータを用いた実験では，Gfarm の使用するファイルディスクリプタ数が TSUBAME のジョブスケジューラで設定されている上限値に達してしまう問題により，ファイル集約タスクにおいてエラーが発生したため，相同性検索ステップにおける実行結果のみ示す．なお，クエリに用いた FASTA ファイルのシーケンス数は 587,335,484(70[GB] のデータ)，DB に用いた FASTA ファイルのシーケンス数は 106,867,961 である．DB ファイルとして用いているデータは，National Center for Biotechnology Information の提供する nr(non-redudant protein) であり，これは複数のデータベースに収録されているアミノ酸配列のコレクションである．また，クエリには HMP(Human Microbiome Project) によって解析されたデータから得られたヒト口腔内メタゲノムデータ (デンタルプラーク) を用いる．

本実験では，180 ノードの内 1 ノードで Gfarm のメタデータサーバと Pwrake のマスターを動作させ，残りの 179 ノードを Gfarm のファイルサーバと Pwrake のワーカとして動作させる．

実験の流れは以下の通りである．

(1) bzip2 で圧縮されているクエリファイルを解凍し，解凍後のファイルをファイル分割プログラムを用いて 1 ファイル当たり 850,000 シーケンスになるように分割 (nr は 18 ファイルに分割し，GHOSTZ-GPU によって DB にビルド)

(2) 179 ノードにクエリをバランス良く配置 (順番に割り当て) し，DB に関しては 18 ファイルそれぞれに対して複製を 10 ずつ作成 (全ノードのローカルストレージに DB を保持させる)

(3) 提案システムによる相同性検索ステップ (GHOSTZ-

GPU を並列実行)

(4) 各 GHOSTZ-GPU の出力結果ファイルを，ファイル分割前のクエリ単位で集約

70[GB] のクエリを用いた実験結果として，各ノードにおけるタスク (プロセス) レベルにおける並列度の推移を**図 7** に，スレッドレベルにおける並列度の推移を**図 8** に示す．また，250[GB] のクエリを用いた実験結果として，タスクレベル並列度の推移を**図 9** に示す．

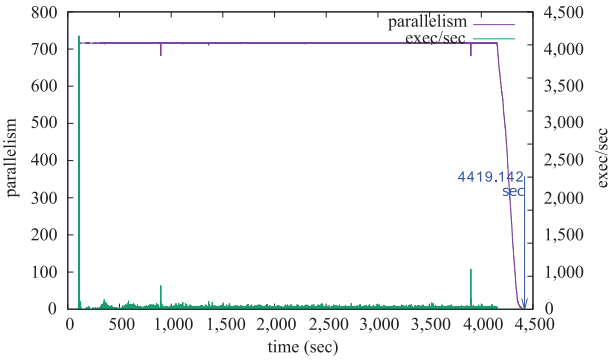

図 7 プロセスレベル並列度 (クエリ:70[GB]

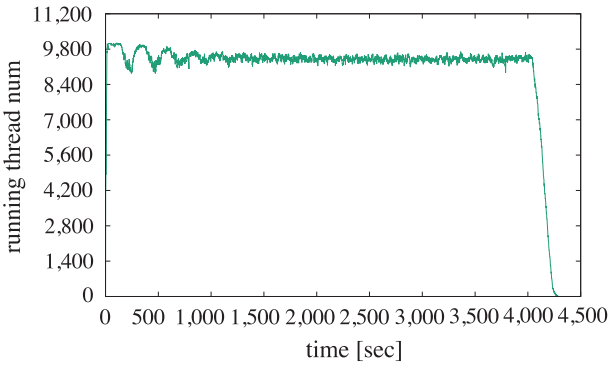

図 8 スレッドレベル並列度

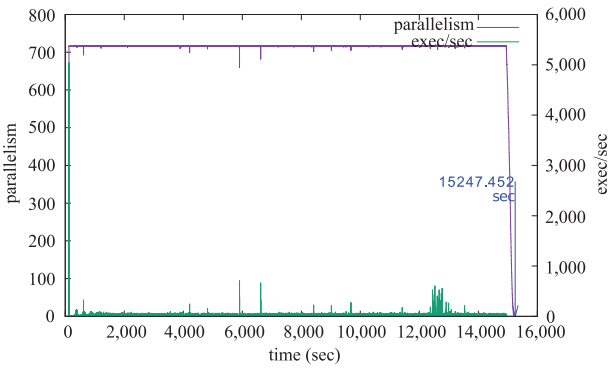

図 9 プロセスレベル並列度 (クエリ:250[GB]

なお，相同性検索ステップにおいては，1 ノード当たりのタスク実行コア数は 4 であり 1 タスクで 14 スレッドを用いて相同性検索を実行している．結果より，並列度はタスクレベル (プロセスレベル) で 716 であり，スレッドレベルでは最大 10,024(平均 9380) 並列で相同性検索が実行されていることがわかる．また，システム全体の実行時間は 4419.14[s] であり，合計タスク数は 12690 個，1 プロセス (1 タスク) 当たりの平均実行時間は 233[s] であった．なお，1 タスク当たりの最大実行時間は 417[s]，最小実行時間は 47[s]，標準偏差は 37 であった．1 タスクあたりの平均実行時間は，18 ノードで行った事前実験における実行時間とほぼ同じであり，リモートノードへの読み込み/書き込みによるネットワークの輻輳が発生せずにローカル読み込み/書き込みを活かすというシステムの特徴を発揮できていることを示している．実際に，各タスクにおいて DB に関してローカル読み込みが行われていることを実行ログより確認している．最後に，相同性検索ステップにおけるシステム全体のファイル読み込みバンド幅の推移を**図 10** に示す．なお，バンド幅の導出は GHOSTZ-GPU のソースコード内のすべてのファイル read/write 命令の前後に gettimeofday() を利用したタイマーを挿入し，読み込み/書き込みバイト数を経過時間で割った値を，積み上げグラフの形で全タスク分足し合わせることで求めている．

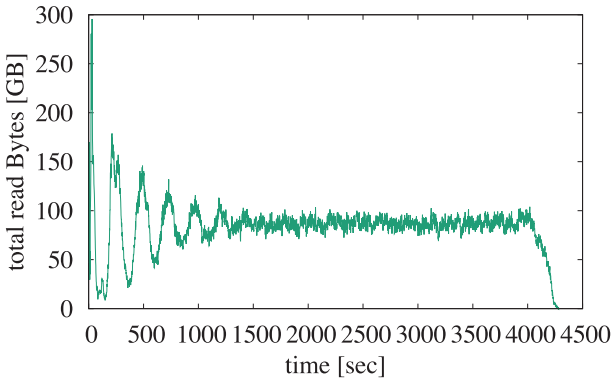

図 10 相同性検索ステップにおける合計読み込みバイト数

5.1.2 ファイル集約タスク

次に，相同性検索ステップにおいて出力された結果ファイルをクエリ毎に集約するタスクの実験結果について述べる．クエリに 70[GB] のものを用いた実験おけるプロセスレベルの並列度を**図 11** に示す．

ファイル集約タスクは図 3 に示した通り，木構造の形のタスク依存関係になっているため並列度はタスク開始時点で最も高く時間が進むに連れて小さくなっている．

5.2 相同性検索結果ファイルの比較

提案システムは，大規模な 1 つの DB ファイルを分割して相同性検索を実行することで，メモリの枯渇問題を防ぐことが可能になっている．しかし，GHOSTZ は DB をクラスタリングする特徴があるため，DB ファイルの分割した場合としない場合でクラスタの代表点が異なり，出力結

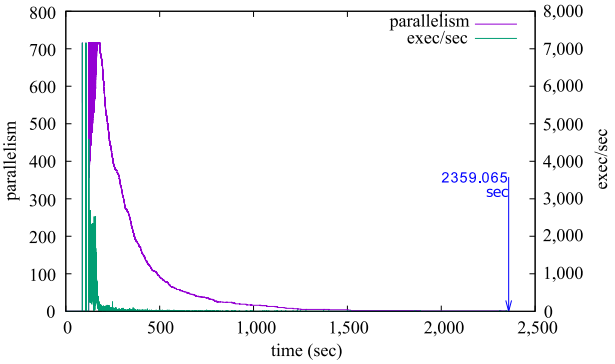

図 11 ファイル集約タスクにおけるタスクレベルの並列度

表 2 1GB の DB を用いた実験の比較結果

DB の分割数	出力サイズ	オリジナルの結果に対する類似度
1	680.4 MB	99.999 %
2	1.1 GB	96.931 %
3	1.4 GB	96.876 %
4	1.8 GB	97.012 %
5	2.4 GB	97.218 %
6	2.5 GB	97.355 %
7	3.1 GB	97.546 %
8	3.4 GB	97.594 %
9	3.7 GB	97.770 %
10	4.2 GB	97.851 %

果も異なってしまう可能性がある．

本節では，この DB 分割による出力への影響がどれほど大きいか調査した結果を示す．まず初めに，テストデータとして 100MB のクエリと DB ファイルに対して，提案システムにおける DB の分割数を 1 から 10 に変化させて実行した出力結果と，単一ホストで動作する GHOSTZ を用いて DB を分割せずに実行した結果との比較を行った．なお，比較にあたって，ある結果 A のレコードの中でもう一方の結果 B にも現れているレコードの B の全体のレコードに対する割合を 2 つの結果の類似度として求めるプログラムを Spark で実装し，それを評価に用いた．今回の場合，このファイル A は DB 分割を用いる提案システムの結果を指し，ファイル B は非分割の GHOSTZ による結果を示す．

比較結果を**表 2** に示す．表中のサイズはファイル A のサイズを表している．なお，非分割による結果であるファイル B のサイズは 680.4MB であった．

結果より，分割数を 1 にした場合は単一の GHOSTZ によるものとほぼ同じ結果が現れているが，DB の分割数を増やすに連れて，出力ファイルサイズが大きくなっていることがわかる．出力ファイルサイズの肥大化の原因としては，今回の実験で GHOSTZ のパラメータの一つであるクエリ毎の検索結果の最大数を，提案システムと単一の GHOSTZ で同じにしているため，提案システムが最大で DB の分割数倍の大きさのファイルを出力し得る事による．類似度は DB の分割によって少し低くなっていることがわかるが，分割数には依存していない．類似度の低下に関しては，DB の分割によってクラスタが変化することで各レコードのエラー値やスコアも変わり，それに起因して本来結果として現れて欲しいレコードが現れないと考えれる．

表 3 62GB の DB を用いた実験の比較結果

DB の分割数	出力サイズ	オリジナルの結果に対する類似度
18	28.9 GB	66.71 %

表 4 DB のサイズに対する DB 分割の影響に関する実験結果

DB のサイズ	出力サイズ	オリジナルの結果に対する類似度
5 GB	56.8 GB	97.927 %
10 GB	44.6 GB	98.365 %
20 GB	35.2 GB	91.885 %
30 GB	30.1 GB	79.63 %
40 GB	26.4 GB	63.352 %

また，5.1 における実験の一部のクエリに対する結果に関して，同様の比較を行った結果を**表 3** に示す．

結果より，類似度が比較的低い事がわかるが，これは**表 4** からわかるように DB のサイズが大きくなるほど，分割による影響が大きいためであると考えられる．なお，表 4 における実験では，100MB のクエリを用いて，DB の分割数を 18 にして実験を行っている．また，表 4 より，DB のサイズが大きくなるほど出力ファイルサイズが小さくなっているが，これは DB のチャンクサイズが大きくなることでクラスタリングによる効率化の影響も大きくなり，より重要な結果のみが出力に現れているためである．

結論として GHOSTZ を用いた場合 DB の分割による結果への影響は少ないとは言えないが，実行時にメモリ不足に陥らないサイズの DB を用いる場合は，DB を分割せずに Gfarm でその複製をもたせることによって高速に処理可能である．この影響の軽減策の検討や BLAST やその他ツールを用いた場合の比較は今後の課題に含まれる．

5.3 スケーラビリティに関する実験

スケーラビリティに関する実験では，DB の元ファイルとして 3[GB] のものを 1 つ，クエリには合計で約 21[GB] のファイルを 2148 個 (179 ノード× 4 プロセスで 3 ステップ分) 用いて，180，128，64，32，16 ノードにおいてストロングスケーリングに関する評価を行った．なお，DB に関しては全ノードに複製を持たせている．実験結果を**図 12** に示す．

結果より，相同性検索ステップに関してノード数が増えるに従って性能がスケールアウトしていることがわかる．また，前処理 (ファイルの分散配置) に関してはノード数が増えるに従い実行時間が増加しているがシステム全体の処理において相同性検索ステップの割合が最も大きいため，入力ファイルサイズをより大きくすることで，他の処理時

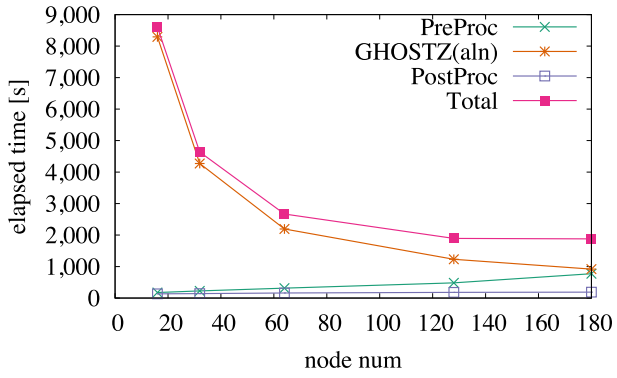

図 12 提案システムのスケーラビリティ

間が隠蔽されシステムのスケーラビリティを高めることができると考えられる．

また，相同性検索ステップにおけるファイル読み込みの合計のバンド幅を**図 13** に示す．

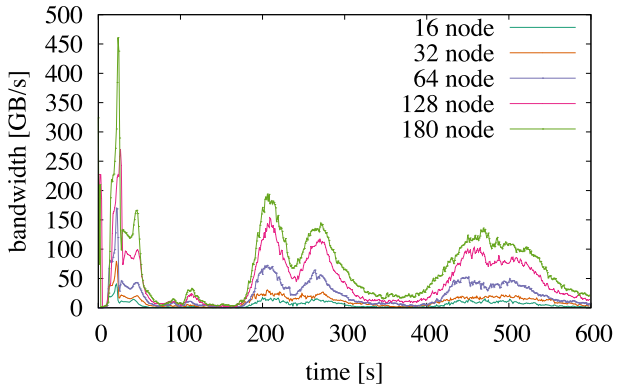

図 13 相同性検索ステップにおける読み込みバンド幅に関するスケーラビリティ

なお，図 13 は，全体で 9,000 秒ある相同性検索ステップのうち最初の 600 秒までを抜粋している．グラフより，最大の合計バンド幅は約 450[GB/s] であり，時間が経過するに連れて各ノードにおけるタスクの実行タイミングの差が大きくなっていきグラフ上の山がなだらかなものになっていくのが見て取れる．また，ノード数の増加に対してバンド幅がスケールしている事がわかる．

6. まとめと今後の課題

本研究では，Gfarm と Pwrake を用いた相同性検索システムについて，TSUBAME3.0 を用いた大規模環境における評価実験を行った．結果として，提案システムは高いスケーラビリティを示し，180 ノードを用いた実験ではビルド済みの約 1 億レコードで構成される non redudant DB と約 5 億レコードから構成されるシーケンスデータに対しておよそ 2 時間で相同性検索を実行した．

今後の課題として，実行時間のモデル化による最適なパラメータ推定やファイル集約タスクの高速化の検討が挙げられる．さらに，ファイル出力結果に関してゲノム解析で利用しやすいような形の検討や，相同性検索の精度を高める方法の検討等も行いたい．

謝辞 本研究の一部は，JSPS 科研費 17H01748，国立研究開発法人新エネルギー・産業技術総合開発機構 (NEDO)，JST CREST JPMJCR1414 および富士通研究所との共同研究によるものである．

なお全ての計算は，東京工業大学のスパコン TSUBAME3.0 を用いて行った．

参考文献

[1] イルミナ社：ヒトの健康における細菌およびメタゲノム，入手先 ⟨https://jp.illumina.com/content/dam/illumina-marketing/apac/japan/documents/pdf/publicaton_metagenome_forhumanhealthj.pdf⟩ (2015)
[2] ヒト腸内メタゲノム解析が広げる医療展開，入手先 ⟨https://www.jstage.jst.go.jp/article/kagakutoseibutsu51/12/51_802/_pdf/char/ja⟩ (2013)
[3] Gaspar Taroncher-Oldenburg, Susan Jones, Martin Blaser, Richard Bonneau, et al, Translating microbiome futures, Nature Biotechnology volume, 36, pp.1037-1042 (2018)
[4] Metagenomics Market Size, Share, Industry Trends Report, 2018-2025, GVR-2-68038-452-9 (2018)
[5] Kanehisa M, Goto S, KEGG: Kyoto Encyclopedia of Genes and Genomes, Nucleic Acids Research, 28(1), pp.27-30 (2000)
[6] Kanehisa M, Goto S, Sato Y, Furumichi M, Tanabe M, KEGG for integration and interpretation of large-scale molecular data sets, Nucleic Acids Research, 40(D1), pp.109-114 (2012)
[7] Tatusov R, Fedorova N, Jackson J, Jacobs A, Kiryutin B, Koonin E, et al. The COG database: an updated version includes eukaryotes, BMC Bioinformatics, 4(1), pp.41–41 (2003)
[8] Stephen F.Altschul, Warren Gish, Webb Miller, Eugene W.Myers , David J.Lipman1Basic：local alignment search tool, Biol, 215, pp.403–410 (1990)
[9] Shuji Suzuki, Masanori Kakuta, Takashi Ishida, and Yutaka Akiyama：GHOSTX: An Improved Sequence Homology Search Algorithm Using a Query Suffix Array and a Database Suffix Array, PLoS One, 9(8):e103833, pp.1–8 (2014)
[10] Shuji Suzuki, Masanori Kakuta, Takashi Ishida and Yutaka Akiyama：Faster sequence homology searches by clustering subsequences, Bioinformatics, 31(8), pp.1183–1190 (2014)
[11] Benjamin Buchfink, Chao Xie & Daniel H. Huson：Fast and Sensitive Protein Alignment using DIAMOND, Nature Methods, 12, pp.59-60 (2015)
[12] 町田健太, 建部修見: Pwrake/Gfarm による分散並列相同性検索システムの提案, 第 162 回ハイパフォーマンスコンピューティング研究発表会, (2017)
[13] AE Darling, L Carey, WC Feng, The design, implementation, and evaluation of mpiBLAST, ClusterWorld Conference&Expo, (2003)
[14] Masanori Kakuta, Shuji Suzuki, Kazuki Izawa, Takashi ishida and Yutaka Akiyama, A Massively Parallel Sequence Similarity Search for Metagenomic Sequencing Data, International Journal of Molecular Sciences, 18(10):2124, pp.1-11 (2017)
[15] A O'Driscoll, V Belogrudov, J Carroll et. al., HBLAST:

Parallelised sequence similarity A Hadoop MapReducable basic local alignment search tool, Journal of Biomedical Informatics Volume 54, pp.58–64 (2015)
[16] Marcelo Rodrigo de Castro, Catherine dos Santos Tostes, et. al., SparkBLAST: scalable BLAST processing using in-memory operations, BMC Bioinformatics, 18:318, pp.1–13 (2017)
[17] Jia Yua, Jochen Blomb, Alexander Sczyrbac, Alexander Goesmann, Rapid protein alignment in the cloud: HAMOND combines fast DIAMOND alignments with Hadoop parallelism, Journal of Biotechnology, 257, pp.58–60 (2017)
[18] Chaojie Zhang, Koichi Shirahata, Shuji Shuzuki, Yutaka Akiyama, Satohsi Matsuoka：Performance Analysis of MapReduce Implementations for High Performance Homology Search, IPSJ SIG Notes, 2014-HPC-147(29), pp.1–7 (2014)
[19] Osamu Tatebe, Kohei Hiraga, Noriyuki Soda：Gfarm Grid File System, New Generation Computing, Ohmsha, Ltd. and Springer, 28(3), pp.257–275 (2010)
[20] Masahiro Tanaka, Osamu Tatebe：Pwrake: A parallel and distributed flexible workflow management tool for wide-area data intensive computing, IProceedings of ACM International Symposium on High Performance Distributed Computing (HPDC), pp.356-359 (2010)
[21] Shuji Suzuki, Masanori Kakuta, Takashi Ishida and Yutaka Akiyama: GPU-Acceleration of Sequence Homology Searches with Database Subsequence Clustering, PLoS ONE, 11(8):e0157338, pp.1–22 (2016)
[22] Tuning YARN,
入手先 ⟨https://www.cloudera.com/documentation/ enterprise/latest/topics/cdh_ig_yarn_tuning.html⟩