ヒトゲノム解析プログラムGenomon-fusionの大規模スーパーコンピュータ向け最適化と性能モデル化

鈴木 惣一朗[1,a)] 伊東 聰[2,b)] 池田 奈生[3,c)] Balazs Gerofi[1,d)] 三吉 郁夫[3,e)] 丸山 直也[1,f)]
滝澤 真一朗[1,g)] 村瀬 洋介[1,h)] 石川 裕[1,i)] 宮野 悟[2,j)]

概要：現在開発中のポスト「京」スーパーコンピュータのターゲットアプリケーションとして，ヒトのがん遺伝子検出プログラム Genomon-fusion の京コンピュータ上でのチューニングを行った．RNA シークエンスデータに対する 1 検体ベンチマーク計算の実行性能を基に，複数検体同時解析実行におけるスループット性能を予測する性能モデルを構築し，京コンピュータ全ノードを使用した計算における 1 日あたりの解析可能検体数を評価した．また，同性能モデルを用いて，ポスト「京」完成時 2020 年に予想される問題規模での全ゲノムシークエンス解析における実行性能の見積もりを行った．

Tuning and performance modeling of a human gene analysis program for large-scale supercomputers

Soichiro Suzuki[1,a)] Satoshi Ito[2,b)] Nao Ikeda[3,c)] Balazs Gerofi[1,d)] Ikuo Miyoshi[3,e)]
Naoya Maruyama[1,f)] Shinichiro Takizawa[1,g)] Yohsuke Murase[1,h)] Yutaka Ishikawa[1,i)]
Satoru Miyano[2,j)]

Abstract: This paper presents a performance analysis and tuning of a human gene analysis program called Genomon-fusion, one of the priority applications for the post-K supercomputer. We first optimize the analysis workflow running on the K computer using a single RNA sequence sample, which then allows us to build a performance model of multi-sample analyses on parallel computing systems in a streaming way. We present the estimated analysis throughput when the whole system of the K computer is employed, and also discuss its expected performance when applied to whole genome analyses in 2020.

1 理化学研究所 計算科学研究機構, RIKEN AICS
2 東京大学 医科学研究所,
The Institute of Medical Science,The University of Tokyo
3 富士通株式会社, Fujitsu Limited
a) soichiro.suzuki@riken.jp
b) sito@hgc.jp
c) ikeda.nao@jp.fujitsu.com
d) bgerofi@riken.jp
e) miyoshi.ikuo@jp.fujitsu.com
f) nmaruyama@riken.jp
g) shinichiro.takizawa@riken.jp
h) yohsuke.murase@riken.jp
i) yutaka.ishikawa@riken.jp
j) miyano@ims.u-tokyo.ac.jp

1. はじめに

近年の次世代シークエンサ性能向上により，短時間に大量の DNA シークエンスデータを低コストで得ることが可能となった．今後のシークエンサ装置のさらなる性能向上，また，遺伝子診断の日常化などを考慮すると，ヒトゲノム解析の対象となるデータ量は，サンプル数の増加およびサンプルあたりのデータ量の増加により，さらに膨大なものにになるであろう [1]．がんゲノム研究，特に，がん予防の基礎データ解析を目的としたがん遺伝子の網羅的研究（その一例として文献 [2]），そして，個々のがん患者レ

© 2016 Information Processing Society of Japan

ベルでの治療戦略のためには，2020 年において，1 日あたり数千検体の全ゲノムシークエンス解析が必要となると予想される [3]．このような社会的，科学的要求により，京コンピュータクラス，さらにそれ以上の性能を持つ将来のスーパーコンピュータを用いたヒトゲノム解析の必要性が高まっている．一般にスーパーコンピュータシステムは，ハードウェア，システムソフトウェアとも，浮動小数点演算中心の HPC アプリケーション実行を対象に最適化されている．そのような環境下において，整数演算中心のワークフロー型アプリケーションであるゲノム解析プログラムを高効率で実行することは挑戦的な課題である [4], [5]．

以上のような背景の下，ポスト「京」スーパーコンピュータ開発におけるコデザイン・ターゲットアプリケーションのひとつに，東京大学医科学研究所ヒトゲノム解析センターで開発中の，がんのヒト全ゲノム関連データ解析プログラム Genomon が採択された．本論文はその作業の第一歩として行った，Genomon シリーズのうち既に京コンピュータへの移植が完了していた Genomon-fusion [6] に対するチューニングと，その実行性能評価について報告する．

今回のチューニングにより，使用ノード数を 2 倍にする条件の下であるが，10 時間かかっていた 1 検体データの解析を 1.5 時間で完了することが可能となった．実行性能評価については，我々は，単に 1 検体の解析の高速実行に重きを置く capability computing の視点だけではなく，与えられた資源を用いて時間内にいかに多くの検体の解析が可能かという capacity computing の視点からの性能評価も重視している．本論文では，複数検体同時パイプライン実行におけるスループット性能を評価するための性能モデルを構築し，それを用いて京コンピュータ全ノードによる RNA シークエンス（RNA-seq）解析時の実行性能を評価し，1 日あたりチューニング前で約 3,000，チューニング後で約 5,000 検体の解析が可能であると推定した．また，現プログラムではまだ実行が不可能な，ポスト「京」開発完了時（2020 年）に想定される全ゲノムシークエンス（whole genome sequencing, WGS）解析に対する実行性能予測を行い，1 日あたりチューニング前で 35，チューニング後で 53 検体の解析が可能であると推定した．

なお現在，ポスト「京」開発のターゲットアプリケーションとしての Genomon は，Genomon-fusion とは異なる新規開発版 Genomon に移行することが決定している．新 Genomon においても，ゲノム解析プログラムとしての基本的構造は Genomon-fusion と同様であるため，本論文では，本質的には変わらないと想定される現状のプログラムを用いて評価する．また，シークエンサが設置された外部機関とポスト「京」スーパーコンピュータを高速ネットワークで接続し，シークエンシングから，ゲノムデータ転送，解析，解析結果の返送までを連携して行う計画も提案されている．ただし，本論文では，解析プログラム実行環境に既に必要なゲノムデータが全てそろった段階からの実行性能評価を行うものとしている．

2. がんのヒト全ゲノム関連データ解析プログラム Genomon

がんのヒト全ゲノム関連データ解析プログラム Genomon には目的別に複数のプログラムが存在する．我々は，既にオリジナルプログラム開発者により京コンピュータへの移植が完了していた Genomon-fusion を対象として実行性能の評価とチューニングを行った．本章では，Genomon-fusion とその京コンピュータへの移植作業について紹介する．

2.1 Genomon-fusion

Genomon-fusion は，ヒトのトランスクリプトーム（transcriptome，細胞中の mRNA）から得られたシークエンスデータ（RNA-seq）を入力とし，がんの原因となる融合遺伝子というタイプの変異を検出するプログラムである [6], [7], [8]．mRNA から転写された DNA 塩基配列の断片（read, リード）の両端から指定長（リード長）だけ読み取ったペアードエンド（paired end）塩基配列情報を，複数のリードに対して読み取り方向別に 2 ファイルにまとめたものが入力データとなる．融合遺伝子は，異なる遺伝子が連結されて生じた正常には存在しない遺伝子である．Genomon-fusion の最終出力結果は，検出された融合遺伝子のリストで，各融合遺伝子のデータとして，それを構成する融合前の各遺伝子とその切断点（junction）情報などが含まれる．プログラム全体は，アライメント計算，融合遺伝子検出，融合遺伝子検証を行うの 3 つのモジュールからなる（**図 1**）．

アライメントモジュール（Alignment module） まず，アライメントプログラム Bowtie [9] により，各入力リードの参照トランスクリプトーム配列へのマッピング（参照塩基配列内での入力リードの位置同定）を行う．次に，マッピングされなかったリードに対して別のアライメントプログラム Blat [10], [11] により参照ゲノム配列（hg19 [12]）へのマッピングを行う．Blat には部分アライメント機能があり，リード中のアライメントされなかった部分はソフトクリッピングとしてマスクされ，マッピング結果とともにマスク情報も出力される．Genomon-fusion ではこのソフトクリッピング機能を利用して，融合遺伝子の検出を行っている．アライメントモジュールでは，マッピング結果に対して，リードのマッピング座標（どの染色体のどの位置にマッピングされたか）によるソート，重複リードの除去，マッピング結果へのランダムアクセスを高速に行うためのインデックス付けまでの処理を行う．

検出モジュール（Fusion candidate listing module）

© 2016 Information Processing Society of Japan

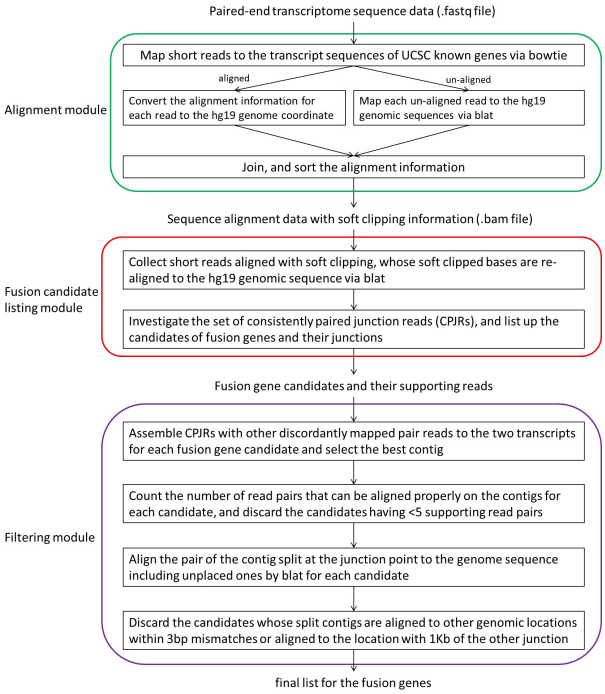


図 1 Genomon-fusion 処理内容

ソフトクリッピングされたマスク領域に対して再度 Blat によるマッピングを行い，二つの異なる遺伝子にまたがってマッピングされたリードを集計して，融合遺伝子である可能性がある候補リストを作成する．

検証モジュール（Filtering module） ここでは，偽陽性（false positive）を減らすために，各融合遺伝子候補に対していくつかの検証計算によるフィルタリングを行い，最終結果として融合遺伝子のリストを出力する．

Genomon-fusion は，入力データを加工して出力する小規模なプログラムをパイプライン的につなげることによりゲノム解析を行うワークフロー型のアプリケーションである．パイプラインを構成する個々のプログラムは，C，C++，Java，Perl などで記述されている．入力データサイズは 2 ファイルの合計で 30〜50GB である．パイプラインを流れるデータ量は解析が進むにつれて小さくなり，アライメント計算後は 8〜10GB，最終解析結果は数 100KB 程度になる．

Genomon-fusion は，東京大学医科学研究所ヒトゲノム解析センターの HGC スーパーコンピュータシステム [13] での実行を前提に開発された．プログラムは，HGC スーパーコンピュータシステム上で稼働しているグリッドエンジン（Sun Grid Engine/Univa Grid Engine）のアレイジョブ（array job）機能とホールドジョブ（hold job）機能を利用して並列化されている．Genomon-fusion パイプラインは，メインシェルスクリプトからグリッドエンジンを経由して，外部プログラムや他のシェルスクリプトをジョブとして実行する形式で記述されている．この時，並列実行可能なジョブはアレイジョブとして投入し，またジョブ間に入出力ファイルを通して実行順に依存関係のあるものについては，ホールドジョブ機能を用いてその実行順序を制御している．

2.2 GFK とそのプログラム構成

Genomon-fusion for K (GFK) [14] は，Genomon-fusion を京コンピュータに移植したものである．移植にあたり，CPU アーキテクチャの相違，利用可能言語環境の相違，ジョブ管理システムの相違，の 3 点からプログラムを改良する必要があった．

京コンピュータの SPARC64 アーキテクチャは，ビッグエンディアンであり，またメモリ上のデータアライメントが x86_64 アーキテクチャと比べて厳格であるため，Genomon-fusion で使われているいくつかの外部プログラムについては，これらに対応するためにソースコードレベルでの改良が必要となった．

また，京コンピュータの計算ノードでは Java 実行環境が提供されていないため，Java で記述された外部プログラムについては，同等な機能を持つ C プログラムで代用する必要があった．

京コンピュータのジョブ管理システムにも，グリッドエンジンのアレイジョブとホールドジョブに相当する機能が存在するが，非並列ジョブを大量に同時実行するには制限があり，Genomon-fusion の並列化方法を変更せずに移植するのは事実上不可能であった．そのため，京コンピュータへの移植では，オリジナルプログラムを並列度を基準として 3 つのバッチジョブに分割し，各バッチジョブでは，オリジナルプログラムのメインシェルスクリプトを，C 言語で作成された MPI プログラムで置き換えた．この MPI プログラムでは，オリジナルプログラムのメインシェルスクリプトからアレイジョブとして実行されていた外部プログラム/シェルスクリプトを，system 関数により各ランク上のサブプロセスとして実行させている．また，グリッドエンジンのホールドジョブ機能により制御していた外部プログラム/シェルスクリプト間の実行順依存関係は，その依存関係を MPI プログラム中に明示的に，必要な箇所では MPI_Barrier による MPI プロセス間の同期を利用して，記述することにより実現している．

オリジナル Genomon-fusion では，メインシェルスクリプトとそこからアレイ/ホールドジョブとして実行される外部プログラム/シェルスクリプトとの間のデータ受け渡しは，全て分散ファイルシステム上のファイルデータにより行っている．GFK においては，同一ランク上でサブプロセスとして実行される外部プログラム/シェルスクリプト間については京コンピュータ計算ノードのローカルディスク上でのファイルデータにより，また，異なるランク上で実行される外部プログラム/シェルスクリプト間についてはファイルシステムの共有領域上のファイルデータによ

© 2016 Information Processing Society of Japan

り，データの受け渡しを行っている．したがって，GFK は 3 つの MPI プログラムで構成されているが，各プログラム中で使用されている MPI 通信機能は外部プログラム/シェルスクリプトの実行順制御のために用いられている MPI_Barrier のみである．MPI プロセス間のデータ受け渡しに，MPI 通信機能を用いず，共有領域上のファイル渡しとしている点については，これにより移植作業が簡便になったが，プログラムの全体性能に対してボトルネックとなる可能性がある．ただし，現在までに京コンピュータを用いて行った GFK の最大規模の計算，30,000MPI プロセスを使用した 50 検体同時計算においても，共有領域上のディスク I/O がボトルネックとなり実行性能が大幅に低下する現象は見られていない．

以下では，GFK を構成する 3 つの MPI プログラム GFKalign, GFKdedup, GFKdetect について説明する．GFK 全体のワークフローについては，付録 A.1 に図示した．

GFKalign　参照用 DNA 塩基配列に対して各リードの位置を同定するアライメント計算を行う．アプリケーション全体で最も計算量が多く，また，次章で示すとおり，アライメントプログラム Blat [10], [11] が実行時間の約 70%を占める．Blat の他に，同じくアライメントプログラムの Bowtie [9]，他に 11 の Perl スクリプトからなる．Embarrassingly parallel であり，入力データの分割により任意プロセス数での MPI 並列計算が可能である．各 MPI プロセス内では，Blat のみが OpenMP によるノード内並列化がなされている．

GFKdedup　アライメント計算結果に対してソート，重複除去，インデックス付けを行う．4 つの Samtools [15] コマンド（view, sort, rmdup, index）から構成される．MPI 並列化もノード内並列化もされていない．

GFKdetect　融合遺伝子の検出を行う．検出計算は 44 の遺伝子領域 [*1] に対して MPI 並列計算を行うが，検出結果の検証計算の一部は並列化されておらず 1 ノード上で計算される．また，ノード内では，OpenMP 化されている Blat を除いて，1 コア実行となる．GFKdetect は，Blat, fasta36, BEDTools, SOAPdenovo-Trans といった C/C++プログラムの他に，33 個の Perl スクリプトからなる．

3 つの MPI プログラム GFKalign, GFKdedup, GFKdetect は，プログラム全体を主に並列度の違いという観点により分割したものである．そのため，オリジナル Genomonfusion を構成する 3 モジュールとの間で，処理内容について若干の相違が見られる．GFKalign はアライメントモジュールのマッピング計算部分にあたる．GFKdedup は，アライメントモジュール中で逐次実行されていたマッピング結果のソート，重複除去，インデックス作成処理部分を抜き出したものに相当する．また，GFKdetect では，検出モジュールと検証モジュールを合わせた処理を行っている．

[*1] 並列数 44 の決定は，DNA 中に存在する生物学的な機能がない，または不明な領域（gap 領域）が基準となっている．gap 領域を境界とすることで，検出領域分割による誤検出を防止している．

表 1　GFK 3 プログラムの特徴

GFKalign	任意プロセス数で MPI 並列実行可能 ノード内は Blat 以外未並列 演算量多い，実行時間 70%が単一プログラム Blat
GFKdedup	逐次実行 ノード内も 1 コア実行 Samtools の 4 コマンド
GFKdetect	検出部は 44MPI，検証計算の一部は逐次実行 ノード内は Blat 以外未並列 多数の Perl スクリプト

表 1 に 3 プログラムの特徴をまとめた．なお，GFK では，Blat の OpenMP スケーラビリティが悪いこと，また，他の外部プログラム/シェルスクリプトにおいてもノード内並列化がなされていないため，1 ノードに 2MPI プロセスを割り当てた実行が想定されている．したがって，使用ノード数は，GFKalign はデータ分割数/2，GFKdedup は 1（逐次実行），GFKdetect は 22 となる．

2.3　既になされているチューニング

我々の作業開始以前に，GFK には，既に以下のチューニングが実施されている [14].

- ホットスポットとなるアライメントプログラム Blat の OpenMP 並列化
- GFKalign における MPI 並列ロードバランス向上のため，サイクリックに入力リードデータを分割

以下本稿では，これらのチューニングがなされた GFK をオリジナル GFK と呼ぶこととする．

3.　チューニング

3.1　プロファイリング

チューニングは，Cancer Cell Line Encyclopedia (CCLE) [16] より入手したリード長 100 bp[*2] × リード数 77,991,623 の RNA-seq データに対するベンチマーク計算を基に行った．入力データファイルは 1 リード情報が 4 行の Fastq 形式のテキストファイルで，リード両エンドに対応した 2 ファイルを，それぞれ 500,000 行（125,000 リード）単位の 624 ファイルに分割した．アルゴリズム的に並列数が可変な GFKalign については，これらを入力ファイルとして，624MPI 並列として実行した．

以下に，京コンピュータにおけるオリジナル GFK による実行プロファイリング結果を示す．**図 2** は GFK 全体に対する 3 プログラム実行時間の割合である．並列化されて

[*2] bp = base pair, そこに含まれる塩基対の個数で表した遺伝子や DNA リードの大きさを示す単位

© 2016 Information Processing Society of Japan

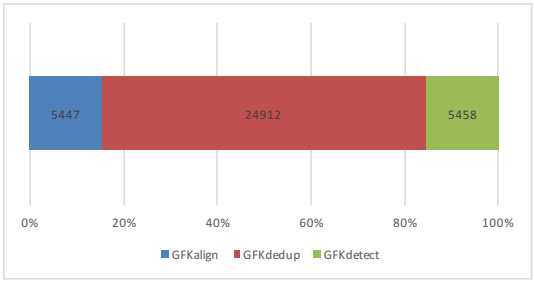


図 2 オリジナル GFK 実行時間（秒）
GFKalign 312 ノード (624MPI)，GFKdedup 1 ノード，GFKdetect 22 ノード (44MPI)

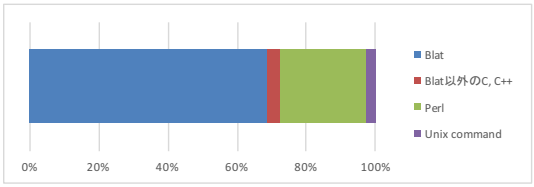


図 3 オリジナル GFKalign 実行時間割合

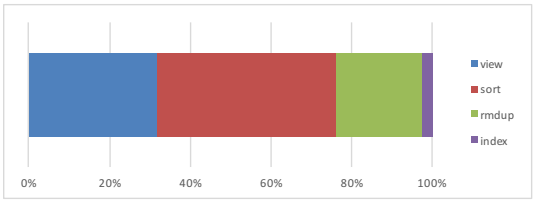


図 4 オリジナル GFKdedup 実行時間割合

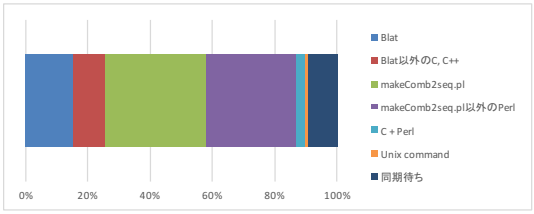


図 5 オリジナル GFKdetect 実行時間割合

いない GFKdedup が実行時間の約 70%を占めていて，この部分の並列化は必須である．なお，MPI プロセスごとの実行時間を積算した値により計算コストを見積もると，GFKalign ≫ GFKdetect > GFKdedup である．

図 3 は，GFKalign 実行での MPI ランク 0 プロセスにおける実行時間を，構成プログラムの言語別に集計したものである．この図では，実行時間の約 70%を占めるホットスポットプログラム Blat の存在を強調するために，Blat のみ個別に集計してある．

図 4 に GFKdedup の構成プログラム別実行時間割合を示す．view, sort, rmdup, index はいずれも Samtools のサブコマンドである．index を除く 3 サブコマンドが実行時間の 97%以上を占めるため，チューニングはこれら 3 サブコマンドの実行を含むワークフロー部分に対して実施する．

図 5 は，GFKdetect 実行での MPI ランク 0 プロセスにおける実行時間を，構成プログラムの言語別に集計したものである．この図では，2 つのホットスポットプログラム Blat（15%），Perl スクリプト makeComb2seq.pl（33%）の存在を強調するために，これらを個別に集計してある．図中の「同期待ち」とは，他 MPI プロセスとのバリア同期実行時間である．

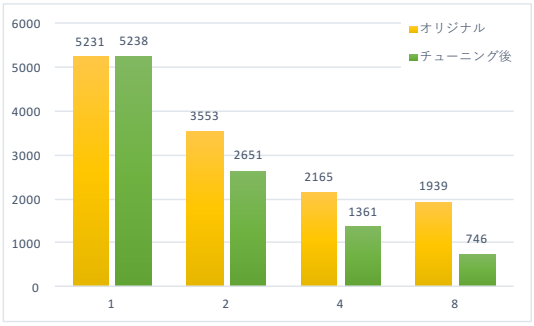


図 6 京コンピュータ，1, 2, 4, 8 スレッドでの リード長 100bp × 26111 リードデータに対する Blat 実行時間（秒）

プロファイリング結果より，以下の方針でチューニングを実施するものとした．まず，GFK 全体のホットスポットとなっている Blat のチューニングを優先する．続いて，アルゴリズム，ワークフローレベルで並列性があると判明した箇所については，並列化により，遊休コア・ノードの削減や負荷分散を改善する．また，Blat 以外のホットスポットプログラムについても，必要に応じてプログラム内部の個別チューニングを行うものとした．

3.2 Blat

GFK 全体をとおしてのホットスポットとなるアライメントプログラム Blat については，OpenMP スケーラビリティの向上を目標にチューニングを実施した．まず，(1) 入力リード単位でアライメント処理を行うメインループは，OpenMP ループ分割されていたが，ロードインバランスが顕著であったため，スレッドスケジューリングをブロックからチャンクサイズ 10 のダイナミックへと変更した．さらに，(2) 並列化がなされていなかった前処理部分を OpenMP 並列化した．また，京コンピュータ/FX100 環境におけるマルチスレッド実行時では，ヒープメモリ要求処理内でロックが多発し，これが計算速度低下の原因となっていたため，(3) 環境変数 “XOS_MMM_L_ARENA_LOCK_TYPE=0” [17] 指定によりスレッドヒープを用いることとし，不要なロックを削減した．**図 6** に，京コンピュータ上における，リード長 100bp × リード数 26111 のテストデータに対する，チューニング前後のコードによる実行時間を示す．本チューニングにより，Blat の OpenMP スケーラビリティは大幅に上昇し，8 スレッド実行時においてオリジナルに対して 2.6 倍の速度向上が得られた．

© 2016 Information Processing Society of Japan

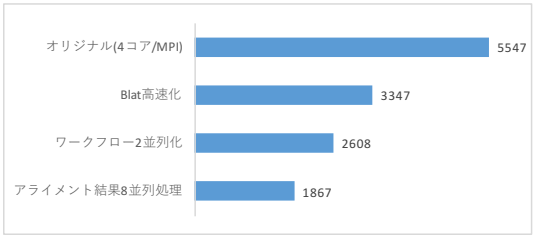


図 7 チューニングによる GFKalign 実行時間（秒）の経過
「オリジナル」は 1 ノード 2MPI プロセス実行（312 ノード），
それ以降は 1 ノード 1MPI プロセス実行（624 ノード）

3.3 GFKalign

オリジナル GFKalign では，Blat 以外はノード内並列化がなされていない．我々はまず，(1) ワークフロー内に，最後のペアードエンド情報の結合処理を除いて，リード情報の読み取り方向別に並列性があることに着目して，最低でも 2 並列実行されるように改良した．また，(2)Blat のスレッド単位で出力された結果ファイルのマージ処理を遅らせ，分散ファイル単位での処理が可能なパイプライン処理についてはコア単位で処理する並列化を行った．なお，オリジナル GFK では Blat の OpenMP 並列化効率が低かったため，GFKalign は京ンピュータ 1 ノードに 2MPI プロセスを割り当てていたが，Blat の OpenMP スケーラビリティが向上したため，1 ノートに 1MPI プロセス実行に改めた．**図 7** にチューニング経過による実行時間の推移を示す．

3.4 GFKdedup

オリジナル GFK では，4 つの Samtools コマンド（view, sort, rmdup, index）を圧縮 BAM ファイル [18] 渡しでパイプライン化し，1 ノード 1 コアを用いた逐次実行となっていた．これに対して，(1) 実行時間の 97%を占める view, sort, rmdup までの処理をアライメント先の染色体単位での処理に MPI 並列化し，26 ノード [*3] で実行するようにした．(2)Samtools コマンド間のデータ移動は，非圧縮 BAM データを Unix パイプで受け渡すようにした．(3)Pthread 化された新しいバージョンの Samtools を用い，ソート処理の一部，データの gzip 圧縮にマルチスレッドを利用するようにした．**図 8** にチューニング経過による実行時間の推移を示す．

3.5 GFKdetect

GFKdetect では，まず，(1) 実行時間の 33%を占める Perl スクリプト makeComb2seq.pl に対して，不必要な処理をループ外に出すなどのチューニングを行った．オリジナル GFKdetect では，Blat 以外はノード内並列化がなされていない．そこで遊休コアの活用のため，(2) ワークフロー

[*3] 26 = 22(常染色体数) + 2(性染色体数) + 1(ミトコンドリア) + 1(未マッチリード)

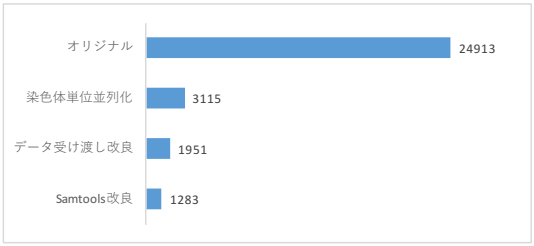


図 8 チューニングによる GFKdedup 実行時間（秒）の経過
「オリジナル」は 1 ノード，それ以降は 26 ノード実行

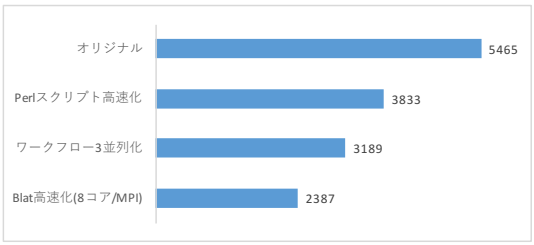


図 9 チューニングによる GFKdetect 実行時間（秒）の経過
「Blat 高速化」のみ 1 ノード 1MPI プロセス実行（44 ノード），それ以前は 1 ノード 2MPI プロセス実行（22 ノード）

中，3 分岐して同時処理可能な箇所を，コア単位で並列実行するようにした．また，GFKalign 同様，Blat の OpenMP スケーラビリティが向上したため，22 ノード上で 44MPI プロセスを実行していたものを，1 ノードに 1MPI プロセスとし，44 ノード実行に改めた．**図 9** にチューニング経過による実行時間の推移を示す．

3.6 単一サンプルを用いたベンチマーク計算

チューニング時に用いたサンプルデータに対する，京コンピュータ上での GFK ベンチマーク計算における実行時間を**表 2** にまとめた．前節で述べたとおり，チューニングによりノード内のコア利用率があがった GFKalign と GFKdetect については，1 ノード 1MPI プロセス実行に改めたため，使用ノード数が 2 倍となっている．また，新たに並列化した GFKdedup では 26 ノードを使用している．表 2 では，参考のため，チューニング後コードに対してオリジナル GFK と同様に 1 ノードに 2MPI プロセス（MPI プロセスあたり 4 コア使用）割り当てた場合での実行時間も示してある．

チューニングによる速度向上比は，GFKalign で 3.0 倍，GFKdedup で 19.4 倍，GFKdetect で 2.3 倍となった．3 プログラムの合計では，6.5 倍の向上となり，使用ノード数は 2 倍となったが，10 時間かかっていた解析が 1.5 時間で可能となった．

同様なベンチマーク計算を，理化学研究所 HOKUSAI-GreatWave システム GPU 演算サーバ（ACSG）[*4] で行った．**表 3** に，ACSG と京コンピュータについて，GFK 実

[*4] GFK の実行には GPU は使用していない．

© 2016 Information Processing Society of Japan

表 2 京コンピュータ 単一サンプルベンチマーク計算
実行時間（秒）と使用ノード数

	オリジナル GFK (4 コア/MPI プロセス)		チューニング後 (4 コア/MPI プロセス)		チューニング後 (8 コア/MPI プロセス)	
GFKalign	5,547	(312 ノード)	3,190	(312 ノード)	1,867	(624 ノード)
GFKdedup	24,913	(1 ノード)	1,339	(13 ノード)	1,283	(26 ノード)
GFKdetect	5,465	(22 ノード)	2,541	(22 ノード)	2,387	(44 ノード)

表 3 GFK 実行に関連する主な CPU 諸元

ACSG	Intel Xeon E5-2670 v3 2.3 GHz (ターボブースト時最大 3.1GHz) コア数 12 メモリバンド幅 68.2 GB/s
京	SPARC64 VIIIfx 2.0 GHz コア数 8 メモリバンド幅 64 GB/s

表 4 ACSG と京コンピュータでの
単一サンプルベンチマーク計算実行時間（秒）

	ACSG	京コンピュータ
GFKalign	377	1,867
(Blat のみ)	289	1,413
GFKdedup	279	1,283
GFKdetect	323	2,387

行に関連する主な CPU 諸元をまとめた．ACSG での GFK 全体実行にはノード数が不足しているため，1 ノード実行で性能評価できるように，(1) 実行されない MPI プロセスのデータは計算済みのダミーデータをあらかじめ用意しておき，(2)GFKdedup と GFKdetect については，京コンピュータでの実行時に MPI 並列化された箇所の実行計算時間が最も長く，同期待ち時間がほぼ 0 であった MPI プロセスを実行して評価した．実際には，ACSG では 1 ノードは 2CPU で構成されているため，1 ノード上に 12 コア実行の 2 つの MPI プロセスを割り当て性能測定を行った．**表 4** に，チューニング後コードによる実行時間を，比較のため京コンピュータ上の結果とともに示す．表中には，GFKalign 中の Blat 実行時間も記した．実行速度比 (ACSG/京) は，GFKalign と GFKdedup で約 5 倍，GFKdetect で 7 倍となった．また，GFKalign の Blat 実行についても約 5 倍となった．これらの速度差が生じた要因については，現在調査を進めている．

4. 性能モデル化による評価

4.1 全体パイプライン実行に対する性能モデル

現在の GFK は，GFKalign, GFKdedup, GFKdetect の 3 つの MPI プログラムからなる．本章では，これら 3 プログラムをひとつのパイプライン（以下，全体パイプライン）にまとめて，複数検体の同時解析（**図 10**）を実施した場合の実行性能についての議論を行う．GFK の全体パイプライン化の実装はまだ行われていないため，ここでは，単一サンプルベンチマーク計算から得られた性能値を基に，全体パイプライン実行性能を予測する性能モデルを構築し，実行性能の指標としてスループットとレイテンシを定義するものとした．

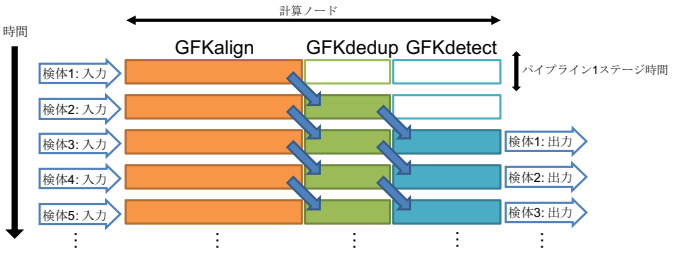


図 10 全体パイプライン実行

前節のベンチマーク計算における GFKalign, GFKdedup, GFKdetect 各ステップでの使用ノード数および実行時間 (秒) をそれぞれ (N_{align}, T_{align}), (N_{dedup}, T_{dedup}), (N_{detect}, T_{detect}) とする．GFKdedup と GFKdetect は，それぞれ染色体数と変異検出遺伝子領域数として，アルゴリズム的に MPI 並列数が固定されているため，全体パイプライン実行においても使用ノード数および実行時間はそのままである．全体パイプラインにおける 1 ステージ実行時間 $T_{pipeline}$ は，GFKdedup と GFKdetect の実行時間の大きい方で律速されるので，それをパイプライン 1 ステージ実行時間 $T_{pipeline}$ とする．

$$T_{pipeline} = \max(T_{dedup}, T_{detect}) \tag{1}$$

一方，GFKalign は任意プロセス数での MPI 並列計算が可能なため，そのノード数を，GFKalign 実行時間が $T_{pipeline}$ 以下になる最小ノード数として決めることができる．したがって，全体パイプラインを構成する全ノード数 $N_{pipeline}$ は次式で定まる．

$$N_{pipeline} = \left\lceil \frac{N_{align}\, T_{align}}{T_{pipeline}} \right\rceil + N_{dedup} + N_{detect} \tag{2}$$

仮に N_{total} ノードの計算機資源が利用可能とすると，同時に $\lfloor N_{total}/N_{pipeline} \rfloor$ 本の全体パイプラインを走らせることができる．その場合，スループット性能 $\mathcal{T}$ は，単位時間あたりの解析可能検体数として評価するものとする．

$$\mathcal{T} = \left\lfloor \frac{N_{total}}{N_{pipeline}} \right\rfloor \frac{1}{T_{pipeline}} \tag{3}$$

1 ステージ実行時間 $T_{pipeline}$ を 3 倍した値は，図 10 に

© 2016 Information Processing Society of Japan

表 5 「京」全ノードを用いた複数検体パイプライン実行

	オリジナル GFK	チューニング後
1 ステージ実行時間 (秒)	24,912	2,387
ノード数/パイプライン	93	558
パイプライン数	891	148
解析検体数/日	3,090	5,356

示すとおり，全体パイプライン計算を開始してから最初の検体データに対する結果が出力されるまでの時間に相当し，全体パイプライン実行におけるレイテンシ $\mathcal{L}$ と解釈できる．

$$\mathcal{L} = 3\,T_{pipeline} \tag{4}$$

なお，上記性能モデルでは，全体パイプライン化によるオーバヘッドの他に，大規模実行時のファイル I/O に関するオーバヘッドの影響も考慮されていない．GFK では，ほぼ全ての I/O はローカルストレージ上で行うが，GFKdedup と GFKdetect においては，MPI プロセス間のデータ転送にファイルシステム上の共有領域を利用している．現在まで京コンピュータ上で行われてきた計算規模（〜50 検体同時計算）においては，共有領域に対する I/O が原因で実行速度が大幅に低下する現象は見られていないが，全体パイプラインを多数本同時実行するような場合には，それが実行時間全体にかかわるボトルネックとなる可能性も考えられる．したがって，上記モデルから得られる実行性能予測値については，現在の GFK をそのまま全体パイプライン化した場合の性能上限値と解釈するべきであろう．

また，上記の性能モデルでは，単一サンプルベンチマーク計算での実行性能を基にしているため，個々の検体データの性質に起因する実行時間の増減やロードインバランスの影響が含まれていない．Genomon-fusion の実行時間は，入力となる塩基配列データの内容に強く依存する．入力データの内容はデータ提供元（検体）ごとに異なり，個々の入力データに対して前もって実行時間を予測することは事実上不可能であるため，本論文での性能推定は，全て単一のサンプル入力データでの実測性能を基に行った．今後，性能モデルの精度を上げるためには，統計モデルを導入するなどの改良が必要となろう．参考情報として，付録 A.2 に，オリジナル GFK により京コンピュータ上で行われた，654 検体の解析における実行時間統計を報告した．

4.2 京コンピュータ全系を用いた複数検体パイプライン実行性能

前節の性能モデル式 (1)〜(4) を用いて，京コンピュータ全系 82944 ノードを用いた全体パイプライン計算時における実行性能を評価した．**表 5** にパイプライン構成とスループット性能見積もりを示す．1 日あたり解析可能検体数は，チューニングにより，約 3000 から約 5000 へと，1.7 倍の向上が得られている．また，レイテンシとしての 1 パイプライン実行時間の 3 倍値は，オリジナル GFK で約 21 時間，チューニング後は約 2 時間となり，約 11 倍の向上が得られた．

表 6 ポスト「京」問題（WGS 解析）に対する「京」全ノードを用いた複数検体パイプライン実行

	オリジナル GFK	チューニング後
1 ステージ実行時間 (秒)	1,245,619	238,737
ノード数/パイプライン	162	558
パイプライン数	512	148
解析検体数/日	35	53

4.3 ポスト「京」問題での実行性能予測

Genomon-fusion の解析対象は RNA-seq であるが，ポスト「京」開発完了時（2020 年）では，全ゲノムシークエンス（whole genome sequencing, WGS）の解析を想定している．将来の想定問題である WGS 解析の現在の京コンピュータ上での実行性能について，現 GFK ではまだ計算できないため，ここでは大まかな性能見積もりについてのみ紹介する．

解析対象が RNA-seq から WGS に変わることによる計算処理コストの増加は，およそ 10 倍と見込まれる．さらに，リード長が現在の 2 倍程度に，読み取り深度（depth）が 5 倍程度増加することが予想される．そこで，WGS 解析での計算時間は，RNA-seq 解析の実行時間を基に，GFKalign と GFKdetect は 100（$= 10 \times 2 \times 5$）倍，リード長に依存しない GFKdedup は 50（$= 10 \times 5$）倍して評価するものとした．

性能モデル式 (1)〜(4) に対して，単一サンプルベンチマーク計算の性能実測より得られた T_{align} と T_{detect} を 100 倍，T_{dedup} を 50 倍した値を用い，京コンピュータ全系 82944 ノードを用いた全体パイプライン計算時における実行性能の見積もり結果を**表 6** に示す．スループットとしての 1 日あたりの解析可能検体数は，オリジナル GFK で 35，チューニング後で 53 となった．また，レイテンシとしての 1 ステージ実行時間の 3 倍値は，オリジナル GFK で 43.3 日，チューニング後で 8.3 日となった．

5. ポスト「京」へ向けての課題

5.1 プログラムレベルでの性能向上

ポスト「京」では，前述の WGS 解析を全系を用いて計算した場合，1 日あたり数千検体規模の解析が可能であることが望まれている．そのためには，オリジナル GFK に対して約 30 倍，チューニング後 GFK に対しても約 20 倍の実行性能向上が必要となる．これらの性能向上には，ハードウェアのみならず，ソフトウェア側のさらなる改良が必要である．特に，現在の GFK では一部でのみノード内並列化されているため，今後のメニーコアアーキテクチャ上

© 2016 Information Processing Society of Japan

での実行を前提とすると，ノード内コアの有効活用を目的とした改良が不可欠である．さらに，ホットスポットのSIMD 化など，並列化以外のチューニングとしても，ハードウェア資源の有効活用が必要である．

また，京コンピュータ全ノードを用いた WGS 解析では，1 検体データの入力からその解析結果の出力までに，チューニング後のコードでも 1 週間以上かかると予想された．遺伝子診断などの立場からは，解析結果はデータ入力後 1 日以内に得られることが望まれる．そのためには，変異検出アルゴリズムの改良・変更などを行い，GFKdedup および GFKdetect 相当部分の並列度を高めるなど高速化し，レイテンシとしての 1 パイプライン実行時間 $T_{pipeline}$ の短縮を目指す必要がある．

5.2 ワークフロー型アプリケーション開発実行環境の検討

Genomon-fusion の京コンピュータへの移植および今回のチューニングでは，MPI 並列化およびノード内並列化は「手作業」で実施してきた．しかし，今後のプログラムの複雑化，解析対象データ量の増加，メニーコアアーキテクチャへの対応などを考慮すると，ミドルウェアとしてなんらかの開発実行環境に頼ることが現実的である．ここでは本論文の最後として，大規模スーパーコンピュータ上でのゲノム解析プログラムの高効率実行を実現するための開発実行環境に対する必要要件について議論する．

Genomon-fusion に代表されるゲノム解析プログラムでは，実行パイプライン中に 2 種類の対照的なプログラム群が混在しているのが特徴である．

プログラム群 1

- 演算量: 多
- プログラム数: 少
- I/O: 多（GB 以上）
- Embarrassingly parallel

プログラム群 2

- 演算量: 少
- プログラム数: 多
- I/O: 少
- 可能な並列化は主にワークフローレベル

ここで，プログラム群 1 は主にアライメント計算プログラム，プログラム群 2 はそれ以外のプログラムからなる．Genomon-fusion を含む一部のゲノム解析プログラムの高効率実行を困難にしているのは，実行時間に占める両プログラム群の割合に大きな優劣がないため，同時に両者に最適な計算機資源を割り当てる必要があるためである．プログラム群 1 については単純なノード間並列・ノード内並列で充分であるが，プログラム群 2 を考慮すると，動的負荷分散を前提とした，（パイプラインを構成するプログラム単位での）ジョブ管理を，遊休コア削減のために，コアレベルで行えるのが理想である．さらに，パイプライン中のデータ入出力の多い箇所については，データの局所性への配慮も必要となる．

また，HPC 経験の浅いバイオ研究者にも，チェックポイントやエラー発生時のジョブ単位での自動再実行などの機能を持った，パイプライン処理プログラムを比較的容易に開発できることが理想である．

我々は以上の要件を満たすものとしての開発実行環境の検討を始めている．

6. 関連研究

スーパーコンピュータシステム上での大規模ゲノム解析としては，ノード間並列に Sun Grid Engine などシステム提供のジョブ管理システムを利用したものは文献 [4], [5] や，その参考文献に事例が見られる．また，文献 [5] には，Apache Hadoop [19] や Apache Spark [20] 上に実装されたゲノム解析ワークフロー/パイプライン構築フレームワークが紹介されている．

一方，浮動小数点演算中心の並列 HPC アプリケーション実行に最適化されたスーパーコンピュータシステムでは，ジョブ管理システムの制限などにより，逐次ジョブの同時大量投入によるノード間並列化は事実上不可能な場合が多い．また，Hadoop/Spark が利用可能な環境も少ない．そのため，そのような環境では，MPI ベースの並列化フレームワークを用いて解析パイプラインを実装するか，ゲノム解析プログラム内で自ら MPI 並列化を行う必要がある．

このようなゲノム解析プログラムとして，京コンピュータ上で全ゲノムシークエンス解析を行う NGS Analyzer [21] がある．NGS Analyzer は既存ツールと独自に開発した解析プログラムをシェルスクリプトで連結させたパイプラインとして実装されている．また，文献 [22] では，NGS Analyzer の MPI 並列化部分を京コンピュータ上の MapReduce 実装である K MapReduce (KMR) [23] 上に再構成した報告がある．NGS Analyzer では，全ゲノムシークエンス解析を行っているが，検出対象が一塩基バリアント（SNV, single-nucleotide variant）と挿入/欠失（InDel, insertion/deletion）といった局所的なものであるため，Genomon-fusion における融合遺伝子検出と比較すると，遺伝子座標に対する領域分割並列化が比較的容易となっている．

文献 [24] では，Genomon シリーズのひとつである，エクソーム（exome, タンパク質に翻訳される DNA 領域）シークエンス解析を行う Genomon-exome [25] の，アライメント計算部分のみであるが，京コンピュータへの移植報告がある．ノード間並列化には，類似配列検索ツール GHOST-MP [26] に使用された，MPI により実装された Master-Worker モデルによるタスク並列分散処理フレームワーク MPIDP を用いている．オリジナルコードでグリッドエンジン経由で実行していたジョブを，ほぼそのまま

© 2016 Information Processing Society of Japan

MPIDP のタスクとし実行しているため，GFK の GFKd-edup に相当するタスクが並列化されておらず，使用ノード数を増加させた場合にそこがボトルネックとなっているようである．GFK では，オリジナル Genomon-fusion を並列度により 3 つの MPI プログラムに分割し，並列化された処理を固定された計算ノードに割り当てるという，この文献とは違ったアプローチを採用している．しかし，前章で述べたように，検出検証部分のアルゴリズム複雑化，アルゴリズム変更への対応，ノード間処理のインバランス解消などを考慮すると，我々も今後はなんらかのタスク並列分散処理フレームワークを利用することが現実的であると考えている．

既存の MPI ベースのタスク並列分散処理フレームワークでは，大規模実行時のスケーラビリティを重視したものとして，ADLB(Asynchronous Dynamic Load Balancing) [27], [28] があげられる．ADLB には，それを用いて実装されたワークフロー型プログラム言語 Swift/T [29] もあるが，現在のところ Swift/T を用いたゲノム解析事例はまだないようである．

7. まとめ

本論文では，現在開発中のポスト「京」スーパーコンピュータのターゲットアプリケーションとして，ヒトのがん遺伝子検出プログラム Genomon-fusion の京コンピュータ上でのチューニングを行った．チューニングにより，使用ノード数を 2 倍にする条件の下であるが，10 時間かかっていた 1 検体データの解析を 1.5 時間で完了することが可能となった．性能評価として，複数検体同時パイプライン実行におけるスループット性能を評価するための性能モデルを構築し，それを用いて京コンピュータ全ノードによる，RNA-seq 解析およびポスト「京」開発完了時（2020 年）に想定される WGS 解析に対する実行性能予測を行い，1 日あたりチューニング後コードで，約 5,000 検体（RNA-seq），53 検体（WGS）の解析が可能であると推定した．

今後のチューニングの余地としては，ノード内コアのさらなる活用，ホットスポットの SIMD 化，MPI 並列化アルゴリズムの改良などが考えられる．性能評価モデルについても，本モデルでは考慮されていない，検体単位での実行時間のばらつきや I/O オーバヘッドを含めることなどによる精度向上が必要である．

また，大規模スーパーコンピュータ上でのゲノム解析プログラムの高効率実行を実現するための開発実行環境（ミドルウェア）に対する必要要件について議論した．今後は，ビッグデータをキーワードとしたデータサイエンスとして，ワークフロー型アプリケーションを大規模スーパーコンピュータ上で実行する必要性が増加していくものと考えられ，このような開発実行環境はバイオ以外の分野でも有用なものとなると思われる．

現在，ポスト「京」開発のターゲットアプリケーションとしての Genomon は，Genomon-fusion とは異なる新規開発版 Genomon に移行することが決定している．新 Genomon においても，ゲノム解析プログラムとしての基本的構造は Genomon-fusion と同一であることを確認している．そのため，本研究で得られた成果および経験は，新 Genomon においても有効であろう．

謝辞 本論文の一部は，文部科学省「特定先端大型研究施設運営費等補助金（次世代超高速電子計算機システムの開発・整備等 ）」で実施された内容に基づくものです．本論文の成果の一部は，理化学研究所のスーパーコンピュータ「京」および HOKUSAI-GreatWave システムの利用により得られたものです．

参考文献

[1] Bourne, P. E., Lorsch, J. R. and Green, E. D.: Perspective: Sustaining the big-data ecosystem, *Nature*, Vol. 527, No. 7576, pp. S16–S17 (2015).
[2] Yoshida, K. et al.: The landscape of somatic mutations in Down syndrome-related myeloid disorders, *Nat Genet*, Vol. 45, No. 11, pp. 1293–1299 (2013).
[3] Zuk, O., Schaffner, S. F., Samocha, K., Do, R., Hechter, E., Kathiresan, S., Daly, M. J., Neale, B. M., Sunyaev, S. R. and Lander, E. S.: Searching for missing heritability: Designing rare variant association studies, *Proceedings of the National Academy of Sciences*, Vol. 111, No. 4, pp. E455–E464 (2014).
[4] Puckelwartz, M. J. et al.: Supercomputing for the parallelization of whole genome analysis, *Bioinformatics*, Vol. 30, No. 11, pp. 1508–1513 (2014).
[5] Kovatch, P., Costa, A., Giles, Z., Fluder, E., Cho, H. M. and Mazurkova, S.: Big Omics Data Experience, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, New York, NY, USA, ACM, pp. 39:1–39:12 (2015).
[6] Genomon-fusion: `http://genomon.hgc.jp/rna/`.
[7] 白石友一，伊東 聰：Genomon fusion を用いたトランスクリプトームビックデータ解析プラットフォームの構築，医学のあゆみ，Vol. 249, No. 10, pp. 1125–1130 (2014).
[8] Shiraishi, Y. et al.: Integrated Analysis of Whole Genome and Transcriptome Sequencing Reveals Diverse Transcriptomic Aberrations Driven by Somatic Genomic Changes in Liver Cancers, *PLoS ONE*, Vol. 9, No. 12, p. e114263 (2014).
[9] Bowtie: `http://bowtie-bio.sourceforge.net/index.shtml`.
[10] Blat: `http://genome.ucsc.edu/cgi-bin/hgBlat`.
[11] Kent, W. J.: BLAT—The BLAST-Like Alignment Tool, *Genome Research*, Vol. 12, No. 4, pp. 656–664 (2002).
[12] UCSC Human Genome Browser – hg19: `https://genome.ucsc.edu/cgi-bin/hgGateway?db=hg19`.
[13] 東京大学医科学研究所ヒトゲノム解析センター HGC スーパーコンピュータシステム：`https://supcom.hgc.jp/`.
[14] Ito, S., Shiraishi, Y., Shimamura, T., Chiba, K.-I. and Miyano, S.: High performance computing of a fusion gene detection pipeline on the K computer, *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, pp. 1441–1447 (2015).

© 2016 Information Processing Society of Japan

[15] Samtools: http://samtools.sourceforge.net/.
[16] Broad-Novartis Cancer Cell Line Encyclopedia: http://www.broadinstitute.org/ccle/home.
[17] 理化学研究所 計算科学研究機構：京コンピュータオンラインマニュアル “man lpgparm”.
[18] The SAM/BAM Format Specification Working Group: Sequence Alignment/Map Format Specification, https://samtools.github.io/hts-specs/SAMv1.pdf.
[19] Apache hadoop: http://hadoop.apache.org/.
[20] Apache Spark: http://spark.apache.org/.
[21] NGS Analyzer: http://www.csrp.riken.jp/application_d_e.html#D2.
[22] 滝澤真一朗，松田元彦，丸山直也：MapReduce による計算科学アプリケーションのワークフロー実行支援，ハイパフォーマンスコンピューティングと計算科学シンポジウム論文集，Vol. 2014, pp. 1–9 (2013).
[23] Matsuda, M., Maruyama, N. and Takizawa, S.: K MapReduce: A Scalable Tool for Data-Processing and Search/Ensemble Applications on Large-Scale Supercomputers, *IEEE Cluster 2013 Conference* (2013).
[24] 青山健人，角田将典，松崎由理，石田貴士，秋山 泰：エクソーム解析パイプラインの京コンピュータ上での大規模並列化，情報処理学会研究報告バイオ情報学（BIO），Vol. 2014, No. 33, pp. 1–7 (2014).
[25] Genomon-exome: http://genomon.hgc.jp/exome/.
[26] GHOST-MP: http://www.bi.cs.titech.ac.jp/ghostmp/.
[27] Lusk, E. L., Pieper, S. C., Butler, R. M. and Univ., M. T. S.: More scalability, less pain: A simple programming model and its implementation for extreme computing., *SciDAC Rev.*, Vol. 17, No. 2010 (2010).
[28] ADLB: https://www.cs.mtsu.edu/~rbutler/adlb/.
[29] Swift/T: http://swift-lang.org/Swift-T/.

付　　録

A.1 Genomon-fusion for K (GFK) ワークフロー

図 A·1, **図 A·2**, **図 A·3** に Genomon-fusion for K (GFK) のワークフローを示す．

A.2 検体データによる実行時間のばらつき

ここでは，京コンピュータで行われた 654 検体に対するオリジナル GFK 計算の実行時間統計について報告する．使用したデータはいずれも，単一検体ベンチマークで使用したサンプルデータと同様，CCLE [16] より入手したものである．**図 A·4**，**図 A·5**，**図 A·6** および**表 A·1** に，MPI プロセスごとの実行時間を総和した，各プログラムの実行時間積算値（秒）の統計情報をまとめた．表中の「サンプルデータ」は，4 章の性能モデルによる性能予測に用いた単一検体ベンチマーク計算における積算値である．これらの値は，図中では赤矢印で示してある．標準偏差を平均値で割った相対標準偏差は，それぞれ，0.29, 0.23, 0.24 となった．3 プログラムとも，サンプルデータでの実行時間積算値は平均値前後の標準偏差内に収まっている．

表 A·1　積算実行時間統計（654 検体データ）

	GFKalign	GFKdedup	GFKdetect
最大	9.2×10^6	3.5×10^4	4.4×10^5
最小	2.1×10^6	15	0.3×10^5
平均	4.7×10^6	2.4×10^4	2.5×10^5
標準偏差	1.4×10^6	0.55×10^4	0.61×10^5
サンプルデータ	3.5×10^6	2.5×10^4	2.1×10^5

© 2016 Information Processing Society of Japan

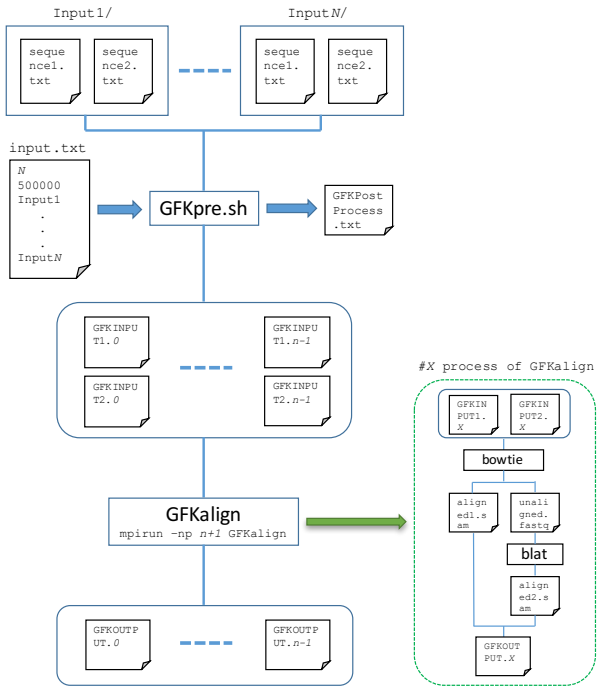


図 A·1 GFKalign ワークフロー

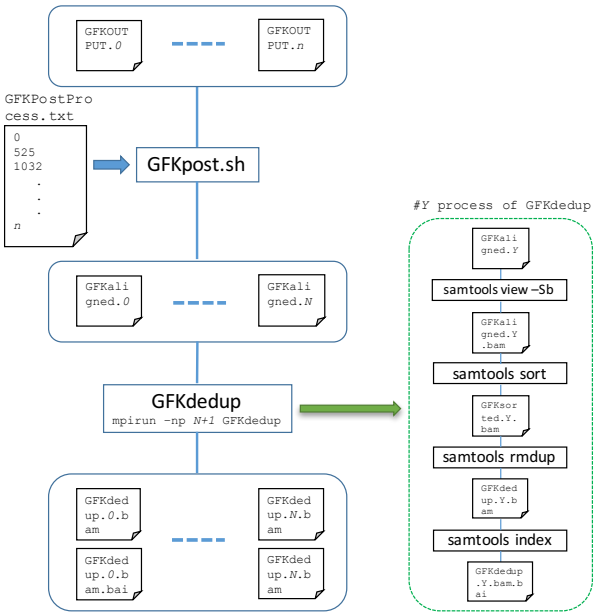


図 A·2 GFKdedup ワークフロー

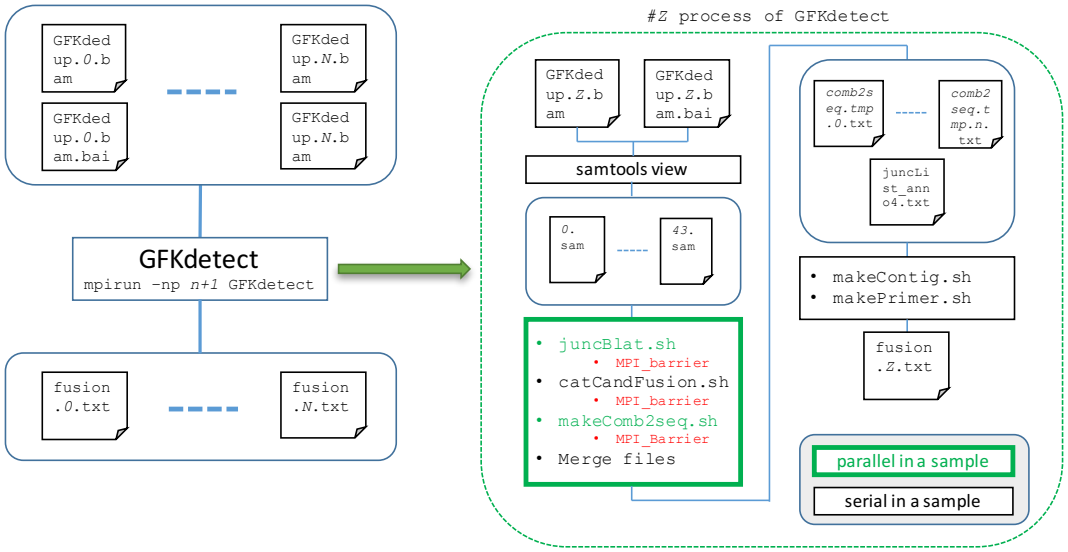


図 A·3 GFKdetect ワークフロー

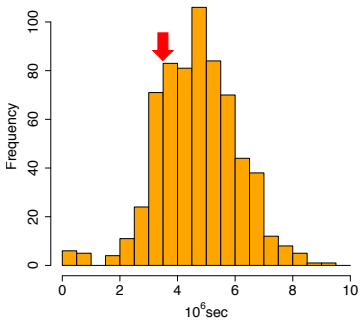


図 A·4 GFKalign 積算実行時間統計

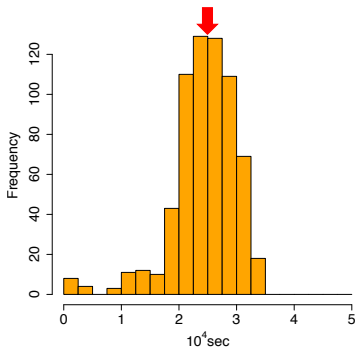


図 A·5 GFKdedup 積算実行時間統計

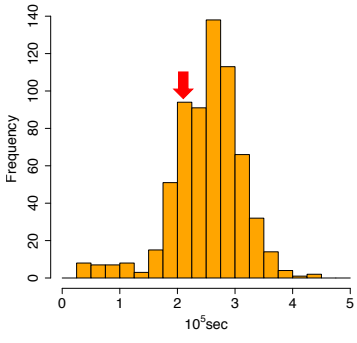


図 A·6 GFKdetect 積算実行時間統計

© 2016 Information Processing Society of Japan