Vol.2022-HPC-185 No.22
2022/7/29

タンパク質構造解析システム AlphaFold の実行時ファイルステージングを用いた高速化

大沢 泰生[1,a)]　遠藤 敏夫[2,1]　野村 哲弘[2]

概要：
スーパーコンピュータ上でビッグデータアプリケーションの実行時間を短くする手法の中に，アクセスするデータを予めローカルストレージにコピーするファイルステージングがある．本論文では，コピーするデータを選択することで更なる高速化を図る異種ストレージ領域を活用したファイルステージングを提案する．この手法を用いることで Lustre ファイルシステム HDD ストレージ上にデータファイルが配置されている共有計算機での AlphaFold の実行において最大 25.2%の高速化に成功した．
アプリケーション実行と提案ステージングでのファイルステージングを並列に実行する実行時ファイルステージングにより，実行時ファイルステージングなしでの実行時間と比較して AlphaFold 実行時間全体で 5.03%の高速化に成功した．
推定するタンパク質によって HHblits 処理でのファイルアクセスのボトルネックとなるファイルが変化することがわかった．

キーワード：AlphaFold，ファイルステージング

1. はじめに

AlphaFold は DeepMind Technologies によって開発されたタンパク質構造予測プログラムである [1]．AlphaFold の主な特徴として，大規模なデータベースと深層学習を活用している点が挙げられる．本学のスーパーコンピュータ TSUBAME3.0 上で AlphaFold を実行することができるが，AlphaFold の実行にあたって容量約 2.3TB のデータディレクトリがリモートストレージの Lustre 上に配置されていることにより，ファイルアクセスがボトルネックとなり，実行速度の低下が発生している．

本論文では，予めアクセスするデータファイルをローカルストレージにコピーするファイルステージングを用いて AlphaFold を実行することでボトルネックを解消することによる高速化を図った．ファイルステージングにあたりデータディレクトリの配置を最適化する必要があることを示し，異種ストレージにデータディレクトリを配置するステージングを提案した．

[1] 東京工業大学情報理工学院
School of Computing, Tokyo Institute of Technology
[2] 東京工業大学学術国際情報センター
Global Scientific Information and Computing Center, Tokyo Institute of Technology
[a)] osawa.t.ag@m.titech.ac.jp

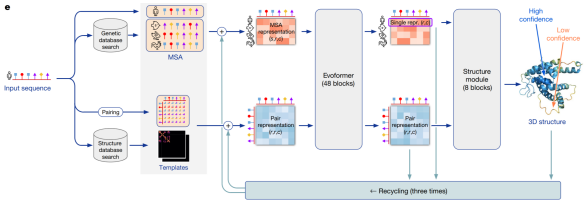

図 1　AlphaFold モデル構造 ([1] より引用)

2. 背景

2.1 AlphaFold2

AlphaFold は DeepMind Technologies によって開発されたタンパク質構造予測プログラムである [1]．AlphaFold の主な特徴として，大規模なデータベースと深層学習を活用している点が挙げられる．AlphaFold の実行内容について説明する．まずタンパク質の塩基配列を入力し，その塩基配列に似たタンパク質の特徴を検索するプログラムである jackHMMer，HHsearch，HHblits を用いて，相同なタンパク質の塩基配列を並べたマルチプルシーケンスアラインメント（以下 MSA）を作成する [2], [3]．その後事前に学習されたデータセットを用いた推論処理により，予測されるタンパク質の立体構造を出力する．AlphaFold のモデル構造を**図 1** に示す．

ⓒ 2022 Information Processing Society of Japan

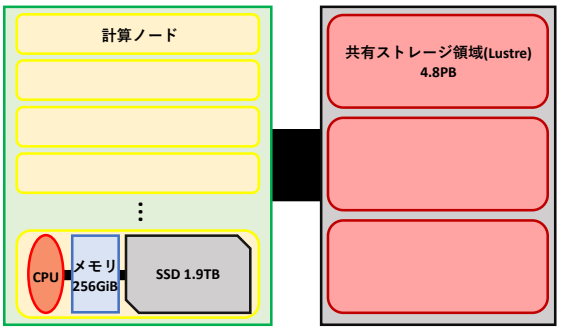

図 2 TSUBAME3.0 ストレージ構造

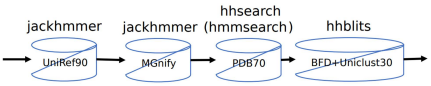

図 4 MSA 取得のパイプライン

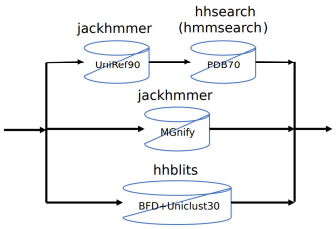

図 5 非同期化した MSA 取得のパイプライン

図 3 MSA 取得処理並列化 ([5] より引用)

2.2 TSUBAME3.0

TSUBAME3.0 は東京工業大学のスーパーコンピュータである．学内外問わず種々の研究・開発部門から利用することが出来る．本論文の全ての実験は TSUBAME3.0 上で実行されている．TSUBAME3.0 は複数の計算ノードを持ち，複数のユーザがそれぞれの計算ノードを利用することで同時に計算を実行することが出来る．各計算ノードにはローカルスクラッチ領域として約 1.9TB の SSD が割り振られている．更に，各計算ノードの CPU には容量 256GiB のメモリがある．計算ノードのストレージ上のデータは各ユーザの利用毎に消去されてしまうので SSD と RAM ディスク上にすべてのデータを常時保存することはできない．そのため，データを保存しておくための大容量ストレージとしてすべての計算ノードからアクセス可能な共有ストレージ領域が存在する．共有ストレージ領域は容量 4.8PB のストレージ 3 つによって構成されている．共有ストレージは HDD で，ファイルシステムとして Lustre を用いている [4]．この共有ストレージに AlphaFold データも配置されている．この AlphaFold データは適切にストライピングされている．

3. 関連研究

3.1 AlphaFold の高速化

AlphaFold を様々な手法で高速化した藤田らの先行研究について述べる [5]．オリジナルの AlphaFold では，MSA

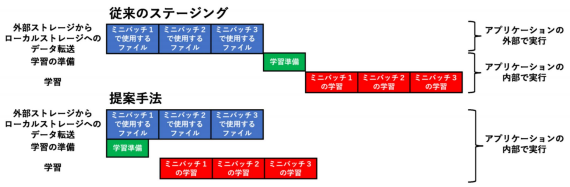

図 1: 従来のステージングと提案手法の違い

図 4 実行時ファイルステージング ([8] より引用)

取得処理に関して 4 つの処理が逐次実行されていたのに対し，これらを並列実行できる 3 つの処理に分けることで高速化に成功したことを示した（**図 3**）．またプロファイリングツール Score-P による AlphaFold のファイルアクセスボトルネック解析がなされた [6]．その結果，HHblits 処理でアクセスする bfd，uniclust30 ディレクトリの内，合計で 20GB ほどのデータファイルにしかほとんどアクセスしていないことがわかった．上記の手法に加え，HHblits の並列実行数の最適化，ソート関数の高速化を施すことで TSUBAME3.0 上での実行時間を平均して 2 分の 1 以下の短縮に成功している．なお実行時間に関して HHblits の高速化が他の結果と比べてうまく高速化されない場合が存在し，CASP14TargetList 内 T1024 に対する実行では HHblits の実行時間は 6000 秒ほどになっている [7]．

3.2 実行時ファイルステージング

アプリケーションがアクセスするデータのファイルステージングとアプリケーション実行そのものを並列して行う手法である [8]．深層学習などファイルへのアクセス時間が大きいアプリケーションに有効な手法である．深層学習を例にすると学習を行うデータセットを複数に分け SSD へとコピーする．アプリケーションによるファイルアクセスが発生した際にコピーされているか確認し，もしコピーが終了していないならばそれがコピーされるまで待つことでファイルのコピーとアプリケーション実行の並列化を可能にしている．

4. AlphaFold の高速化手法

4.1 異種ストレージを用いた高速化

TSUBAME3.0 上で AlphaFold を実行すると，その実行時間の大部分が jackHMMer，HHsearch，HHblits，推論処理によって占められていた．AlphaFold は約 2.3TB の大規模データセットを必要としており，それらのファイルを共有ストレージ領域に格納して実行したため，ファイルステージングによってこの実行時間を短縮することが考えられる．

ファイルステージングはアプリケーションの実行の前にあらかじめアクセスするデータファイルをローカルストレージにコピーすることである．ファイルステージングにより，ファイルのコピー時間がオーバーヘッドになるが高

ⓒ 2022 Information Processing Society of Japan

表 1 各処理がアクセスするデータディレクトリ

処理	データディレクトリ
jackHMMer	uniref90/，mgnify/
HHsearch	pdb70/
HHblits	bfd/，uniclust30/
推論	params/
(オプション)	uniprot/，pdb_seqres/

表 2 全データファイルを Lustre 上/SSD 上に配置した際の各処理の実行時間

	jackHMMer	HHsearch	HHblits	推論	その他
Lustre	718s	567s	294s	430s	120s
SSD	732s	55s	184s	457s	136s

速なファイルアクセスによりアプリケーション実行時間の高速化を図ることが出来る．

すべてのファイルをローカルストレージへとコピーする一般的なステージングでは，ファイルサイズ当たりのファイルアクセス頻度が低いファイルをステージングするとオーバーヘッドによりアプリケーション実行速度が低下してしまう場合がある．そのためファイルステージングを行う場合には，ステージングするファイルを適切に選択する必要がある．

jackHMMer，HHsearch，HHblits，推論の各処理がどのデータディレクトリにアクセスするかを**表 1** に示す．推定するタンパク質が四次構造を持つ複合体の場合，AlphaFold 実行時のオプションで対象が複合体であることをオプションで宣言する．このオプションを追加して実行する場合にアクセスするディレクトリが存在する．今回はすべての実験を四次構造を持たない単量体の設定で実行するので，それらのディレクトリはオプションでのアクセスとして記述している．

全データファイルが Lustre ファイルシステムストレージ上にある場合と SSD 上にある場合で AlphaFold を実行し，jackHMMer，HHsearch，HHblits，推論の各処理の実行時間を比べることで，各処理でファイルアクセスがボトルネックになっているかを確認した．このとき比較する実行時間に SSD へのコピー時間は含まれていない．実験結果を**表 2** に示す．

jackHMMer と推論処理の実行時間に違いは見られなかったが，HHsearch と HHblits 処理は全データが SSD にある場合で実行時間が短縮した．

HHsearch 処理は pdb70 ディレクトリ，HHblits 処理は bfd，uniclust30 ディレクトリにアクセスする．これらのディレクトリと，推論処理がアクセスするコピー時間の短い params ディレクトリをコピーする AlphaFold のステージング実行を提案する．

3.1 節の先行研究により，bfd/cs219.ffdata[*1]へのファイルアクセス時間が非常に多いことがわかっている．[*1]．bfd/cs219.ffdata[*1]と，ファイルサイズが小さいのでコピー時間が数秒で済む bfd/a3m.ffindex[*1]，bfd/cs219.ffindex[*1]，bfd/hhm.ffindex[*1]データすべてをステージングして実行したところコピー時間を含めた実行時間を高速化する結果が得られた．

表 3 提案ステージング：ストレージ別ディレクトリ配置

ストレージ	合計ファイルサイズ	データディレクトリ	ファイルサイズ
Lustre	2183GB	bfd/a3m.ffdata[*1]	1449GB
		bfd/hhm.ffdata[*1]	304.4GB
		mgnify/	64GB
		uniprot/	101GB
		uniref90/	58GB
		pdb_mmcif/	206GB
		pdb_seqres/	0.211GB
SSD	60GB	params/	3.5GB
		pdb70/	56GB
RAM disk	104GB	bfd/a3m.ffindex[*1]	1.692GB
		bfd/cs219.ffdata[*1]	15.66GB
		bfd/cs219.ffindex[*1]	1.573GB
		bfd/hhm.ffindex[*1]	0.123GB
		uniclust30/	86GB

TSUBAME3.0 でのファイルステージングをする場合に，ファイルコピー先として実行ノードのローカルストレージである SSD と，実行ノードの CPU メモリを RAM ディスクとして指定できる．SSD の最大容量は約 1.9TB，RAM ディスクとして用いる CPU メモリの最大容量は 128GB である．ステージングするファイルの容量合計は RAM ディスクの最大容量 128GB を越えているので，すべてのステージングするファイルを RAM ディスクにコピーすることはできない．そのため，RAM ディスクにステージングするファイルを選別し，残りのステージングするファイルを SSD へコピーした．

なお AlphaFold はデータファイルが全て一つのディレクトリに格納されていることを前提としている．この前提と異種ストレージの活用を両立させるため，以下のような実行方式とした．AlphaFold の実行開始時に SSD 上にシンボリックリンクや空ファイルによる疑似データディレクトリを作成する．SSD や RAM ディスクにステージングする場合に空ファイルに対する上書き，ステージング後のシンボリックリンクの再リンクを行う．

4.2 実行時ファイルステージングを用いた高速化

前節では，オリジナルの AlphaFold を対象として，効率的な実行が可能なようなファイルステージング方式を提案した．AlphaFold の実行時間をさらに短縮することを目的に，以下のような改良と組み合わせる．

まず，先行研究 [5], [9] で提案された MSA 取得処理の非同期実行を取り入れる (図 3)．なお先行研究では HHsearch

[*1] bfd ディレクトリ内のファイルは，bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_をファイル名の前に省略している．

表 4　AlphaFold の通常実行とステージング実行の各処理の実行時間

	コピー時間	jackHMMer	HHsearch	HHblits	推論	その他	合計
Lustre		718s	567s	294s	430s	120s	2129s
単純ステージング	2803s	732s	55.0s	184s	457s	136s	4367s
提案ステージング	119s	721s	27.2s	165s	445s	115s	1592s

の後，並列実行できる HHSearch 結果のテンプレート検索が並列実行外で行われていたのでこれを並列実行に含める改良も行った．

上記によって粗粒度な並列化がなされたことを活用し，共有ストレージからのファイルステージングも並列化を行うこととした．これは実行時ファイルステージングに相当し，コピー時間の隠蔽が期待される．具体的には．HHblits の実行前に bfd，uniclust30，params ディレクトリ下ファイルのコピー，UniRef90 への jackHMMer の実行前に pdb70 ディレクトリ下ファイルのコピーを行うようにした．実行時ファイルステージングの実装は subprocess を用いて実装した．

5. 実験と評価

5.1 異種ストレージを用いた高速化

提案ステージングについて，実験を行い評価した．本研究における実験は，TSUBAME3.0 上でプログラムを実行した．AlphaFold のバージョンは 2.1.1 に指定し，AlphaFold への入力配列は PDB エントリー 6MRR で配列長が 68 の配列にした．実行ノード数は通常実行と提案ステージング実行で 1 とした．HHblits 処理の並列実行数を 28 に変更した．

提案するステージング実行の効果の評価を示すため，比較対象として，すべてのデータファイルをステージングする単純ステージングによる AlphaFold の実行時間と比較した．単純ステージングでは，すべてのデータファイルを SSD 上へステージングした．単純ステージング実行において実行ノード数が 1 つでは SSD の容量超過が原因ですべてのファイルをステージングすることができない．そのため，単純ステージング実行での実行ノード数は 2 とした．

提案するステージングについて，実験を行い評価した．AlphaFold の実行において，Lustre 上に全データファイルがある通常の実行，単純ステージングを行った実行，4.1 節で提案するステージングを行った実行の 3 つを比較した．実行結果を**表 4**，**図 5** に示す．

様々な入力配列に対しても，実験を行い評価した．Lustre 上に全データファイルがある通常の実行，4.1 節で提案するステージングを行った実行の 2 つを比較した．実行結果を**図 6** に示す．

HHsearch と HHblits の実行時間が短縮された．具体的には，HHsearch 処理では 88.8%から最大で 95.2%の短縮，HHblits 処理では 0%から最大で 43.9%の短縮に成功した．AlphaFold 実行時間全体の実行時間で 3.09%から最大で 25.2%の短縮化がなされた．

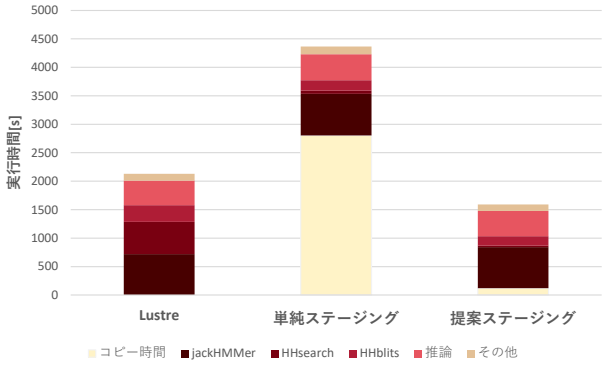

図 5　AlphaFold の通常実行とステージング実行の各処理の実行時間

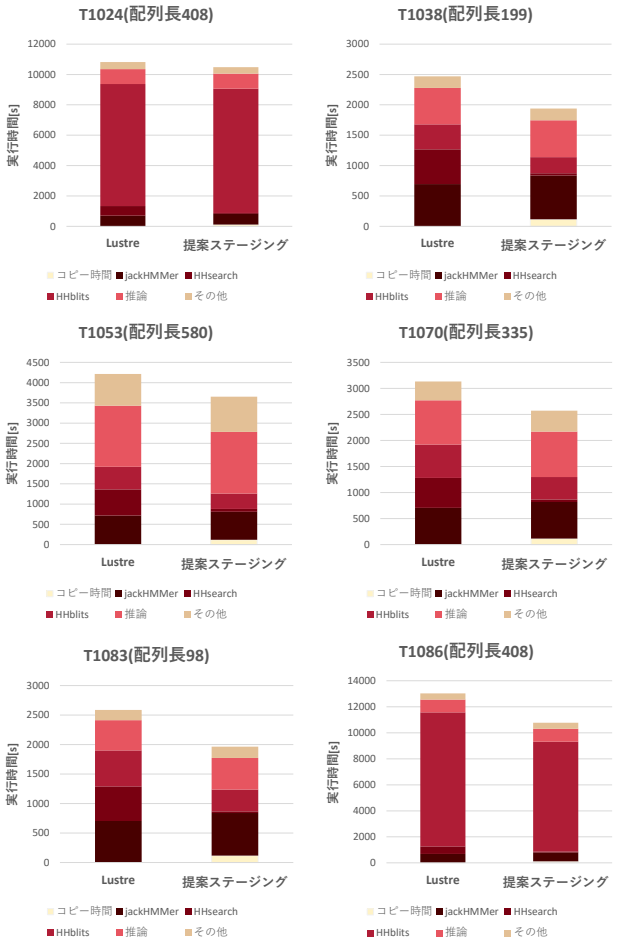

図 6　様々な入力配列に対する AlphaFold の通常実行と提案ステージング実行の各処理の実行時間

5.2 実行時ファイルステージングを用いた高速化

実行時ファイルステージングについて，実験を行い評価した．本研究における実験は，TSUBAME3.0 上でプログラムを実行した．AlphaFold のバージョンは 2.1.1 に指定し，AlphaFold への入力配列は CASP14 での TargetList 内 T1038 で配列長が 199 の配列にした．実行ノード数は先行研究の実行と提案ステージング実行で 1 とした [7]．

実行プログラムコードは TSUBAME3.0 上で利用できる AlphaFold プログラムコードをベースに alphafold/alpha-

表 5 AlphaFold の先行研究での実行と提案ステージングでの実行時ファイルステージング有無実行の各処理の実行時間

	コピー時間	MSA 取得時間	推論	その他	合計
先行研究	38s	930s	600s	203s	1772s
提案ステージング	119s	390s	602s	201s	1311s
提案実行時ステージング		424s	607s	214s	1245s

表 6 MSA 取得処理で並列実行された各処理の実行時間

	MGnify	UniRef90			HHblits		
		コピー時間	UniRef90	HHsearch	合計	コピー時間	HHblits	合計
先行研究	393s		333s	597s	930s		275s	275s
提案ステージング	390s		343s	36.3s	380s		249s	249s
提案実行時ステージング	394s	49.4s	338.4s	36.6s	424s	85.3s	247.3s	333s

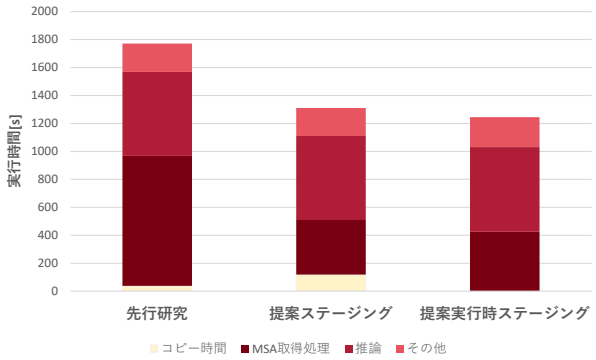

図 7 AlphaFold の先行研究での実行と提案ステージングでの実行時ファイルステージング有無実行の各処理の実行時間

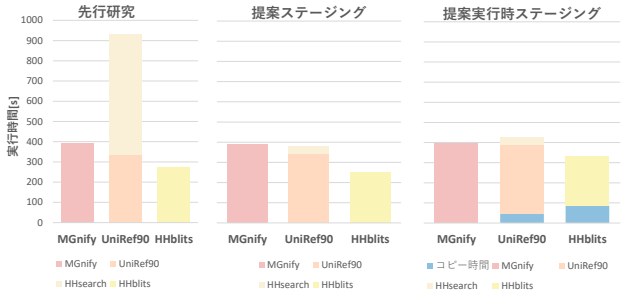

図 8 MSA 取得処理で並列実行された各処理の実行時間

fold/data/pipeline.py と alphafold/alphafold/data/tools/-hhblits.py のみ GitHub で公開されている先行研究のプログラムコードを用いた [5], [9]．pipeline.py に対して concurrent.futures.Executor の max_worker 数を 28 にする変更を加えた．

実行時ファイルステージングについて，実験を行い評価した．AlphaFold の実行において，先行研究に則った bfd, uniclust30 ディレクトリ以下の cs219 ファイル約 25GB をステージングする実行，4.1 節で提案するステージングかつ実行時ファイルステージングを行わない実行，4.1 節で提案するステージングかつ実行時ファイルステージングを行った実行の 3 つを比較した．先行研究に則った実行のみ HHsearch のテンプレート検索については改良前である．実行結果を**表 5**，**図 7** に，MSA 取得処理の実行時間の詳細な内訳を，**表 6**，**図 8** に示す． 提案ステージングでの実行時ファイルステージングの有無の実行時間を比較して，

表 7 T1024 に対する通常実行と様々なステージング実行の各処理の実行時間

	コピー時間	jackHMMer	HHsearch	HHblits	推論	その他	合計
Lustre		700s	631s	8043s	976s	467s	10817s
提案ステージング	118s	734s	56.9s	8185s	988s	431s	10483s
+bfd/hhm.ffdata*1	370s	701s	57.1s	7561s	997s	471s	10157s
+bfd/a3m.ffdata*1	1351s	708s	56.9s	4417s	995s	483s	8010s
+bfd/a3m&hhm.ffdata*1	1469s	698s	85.9s	3611s	997s	474s	7335s

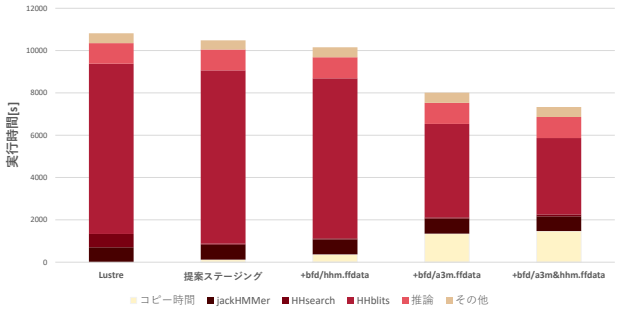

図 9 T1024 に対する通常実行と様々なステージング実行の各処理の実行時間

それぞれのコピー時間を含めた MSA 取得処理時間が短縮された．実行時ファイルステージング無しでの実行では MSA 取得処理のボトルネックになっているのは MGnify に関する処理だったことに対し，実行時ファイルステージング有りでの実行では UniRef90 への jackHMMer 処理がボトルネックになった．実行時間全体で 5.03%の短縮化がなされた．

5.3 T1024 に対する HHblits のファイルアクセスボトルネック

先行研究によると，CASP14TargetList 内 T1024 に対する実行では，HHblits の実行時間が他の結果と比較してうまく高速化されなかった [5]．

本論文でも 5.1 節において，4.1 節で提案するステージングにおける T1024 への実行を行ったが，HHblits の実行時間は改善されなかった．

提案ステージングに加えて，HHblits がアクセスするファイルであるが 4.1 節で提案するステージングファイルには含めなかった bfd/a3m.ffdata*1，bfd/hhm.ffdata*1 をそれぞれ，また，ともに SSD 上にステージングして T1024 に対する実行をしたところ，HHblits 処理実行時間が改善された．

実行条件について，ステージングファイル以外の条件を 5.1 節での実行条件と同じにした．追加する bfd/a3m.ffdata*1，bfd/hhm.ffdata*1 を共にステージングする実行において実行ノード数が 1 つでは SSD の容量超過が原因ですべてのファイルをステージングすることができない．そのため，この場合の実行での実行ノード数は 2 とした．

提案ステージングでの実行時間と提案ステージングに加えて bfd/a3m.ffdata*1，bfd/hhm.ffdata*1 の 2 つのファイルを共にステージングした実行時間を比較することで少

ⓒ 2022 Information Processing Society of Japan

なくともどちらかのファイルアクセスがボトルネックになっていることが分かった．推定するタンパク質によってHHblits 処理において bfd/a3m.ffdata*1，bfd/hhm.ffdata*1 へのファイルアクセスがボトルネックになる場合があると考えられる．

6. まとめと今後の課題

TSUBAME3.0 での AlphaFold の実行を分析した結果，HHsearch と HHblits 処理でファイルアクセスがボトルネックになっていることがわかった．ボトルネックを解消するためにデータファイルを異種ストレージに配置するファイルステージングを行ったところ，入力配列長 68 での実行ではボトルネックは解消され，AlphaFold 実行時間全体では最大 25.2%短縮された．

先行研究の MSA 取得並列処理に対し，実行時ファイルステージングによる高速化により実行時間が短縮された．提案ステージングでの実行時ファイルステージングの有無の実行時間を比較して，それぞれのコピー時間を含めた MSA 取得処理時間が短縮された．実行時間全体で 5.03%の短縮化がなされた．

推定するタンパク質によって HHblits 処理でのファイルアクセスのボトルネックとなるファイルが変化することがわかった．

今後の課題としては，HHblits 処理の様々な入力配列に対する詳細なファイルアクセスパターン解析，コピー時間の改善，AlphaFold に限らないアプリケーションの実行におけるステージングを用いた実行による高速化の可否についての定式化などが挙げられる．

謝辞 本研究の一部は JSPS 科研費 20H04165 の助成によるものです．また本研究は東京工業大学のスーパーコンピュータ TSUBAME3.0 を利用して実施しました．ご助言をいただいた東京工業大学情報理工学院 関嶋政和先生，藤田隼斗さんに感謝いたします．

参考文献

[1] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A. et al.: Highly accurate protein structure prediction with AlphaFold, *Nature*, Vol. 596, No. 7873, pp. 583–589 (2021).
[2] Finn, R. D., Clements, J. and Eddy, S. R.: HMMER web server: interactive sequence similarity searching, *Nucleic acids research*, Vol. 39, No. suppl_2, pp. W29–W37 (2011).
[3] Steinegger, M., Meier, M., Mirdita, M., Vöhringer, H., Haunsberger, S. J. and Söding, J.: HH-suite3 for fast remote homology detection and deep protein annotation, *BMC bioinformatics*, Vol. 20, No. 1, pp. 1–15 (2019).
[4] OpenSFS and EOFS: Lustre, `https://www.lustre.org/`.
[5] 藤田隼斗，野村哲弘，遠藤敏夫，関嶋政和：タンパク質立体構造予測システム AlphaFold の TSUBAME3.0 上での高速化, 情報処理学会研究報告, 2022-HPC-183, No. 3, pp.1-7 (2022).
[6] Knüpfer, A., Rössel, C., Biersdorff, S., Diethelm, K., Eschweiler, D., Geimer, M., Gerndt, M., Lorenz, D., Malony, A., Nagel, W. E. et al.: Score-p: A joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir, *Tools for High Performance Computing 2011*, Springer, pp. 79–91 (2012).
[7] Protein Structure Prediction Center: CASP14 Target List, `https://predictioncenter.org/casp14/targetlist.cgi`.
[8] 樋口遼太郎，三輪忍，八巻隼人，本多弘樹：深層学習における実行時ファイルステージング, 情報処理学会研究報告, 2021-HPC-182, No. 7, pp. 1-8 (2021).
[9] fuji8: fuji8/alphafold, `https://github.com/fuji8/alphafold`.

ⓒ 2022 Information Processing Society of Japan