Vol.2019-HPC-172 No.20
2019/12/19

分散バーストバッファが持つI/Oの性能評価

澤田 一樹[1,a)]　建部 修見[2]

概要：多くのHPCシステムのアプリケーションにおいて，バースト的なI/O性能要求を満たすためにバーストバッファが用いられるようになった．従来の分散ファイルシステムは永続的にデータを保持することが目的とされ，ファイルシステムノードと計算ノードは別ホストに構築されるものとして設計されていた．従って，計算ノードローカルに存在する高速なストレージ資源を用いてファイルシステムを構築する場合，本来持つ高いI/O性能を発揮することができない．一方で，計算ノードローカルのストレージデバイスで一時的に構成する分散バーストバッファに関して，どのようなワークロードとI/Oを持つアプリケーションを高速化できるか明らかになっていない．本研究では，HPC上で実行されうる様々なアプリケーションが持つワークロードに基づいて分散バーストバッファが持つI/O性能を評価することを目的としている．

1. はじめに

多くのHPCシステムのアプリケーションにおいて，バースト的なI/O性能要求を満たすためにバーストバッファが用いられるようになった．従来の分散ファイルシステムは永続的にデータを保持することが目的とされ，ファイルシステムノードと計算ノードは別ホストに構築されるものとして設計されていた．従って，計算ノードローカルに存在する高速なストレージ資源を用いてファイルシステムを構築する場合，本来持つ高いI/O性能を発揮することができない．バーストバッファとは，一時的な作業用の記憶領域として計算ノードローカルに搭載される非常に高速なストレージデバイスであり，現在は全てのシステムにおいてNVMeSSDが用いられている．また，計算ノードローカルのストレージ資源を集約するために分散ファイルシステムを使用する場合もあり，ユーザーにバーストバッファを提供するためにTSUBAME3.0やABCIではBeeOND[1]が採用されている．しかし，計算ノードローカルのストレージデバイスで一時的に構成する分散バーストバッファに関して，どのようなワークロードとI/Oを持つアプリケーションを高速化できるか明らかになっていない．本研究では，HPC上で実行されうる様々なアプリケーションが持つワークロードに基づいて分散バーストバッファが持つI/O性能を評価することを目的としている．

[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)] sawada@hpcs.cs.tsukuba.ac.jp

2. 関連研究

2.1 Gfarm

Gfarmファイルシステム[2]は広域分散ファイルシステムである．広域に性能・容量をスケールアウトすることができ，単一障害点がない．遠隔地のクラスタに渡ってミラーリングを設置することができ，障害・災害に強い．主に研究分野で使用され，HPCI共用ストレージ（～100PB）やすばる望遠鏡データ解析で使用されている．また，ユーザー権限でファイルシステムを構築可能である．バーストバッファのための分散ファイルシステムとしてGfarm/BB[3]が利用可能となっている．

2.2 BeeGFS

BeeGFS[4]は性能を要求される環境下において厳しい負荷に応え，かつ簡単に管理が可能な分散ファイルシステムを目指して開発が開始され，サーバーデーモンはユーザースペースで動作させることができる．導入・保守を簡単にするためにインストーラや管理・監視作業を行うためのGUIアプリケーションも提供されている．また，BeeOND(BeeGFS On Demand)により，単一のコマンドによってBeeGFSインスタンスを作成・停止することができる．

2.3 Octopus

Octopus[5]では従来は厳密に区分されていた，ファイルシステムとネットワークレイヤを再設計し，非常に高速なストレージである不揮発性メモリとリモートサーバーから

メモリに直接アクセスすることにより，低遅延のネットワークアクセスを可能にする RDMA（Remote Direct Memory Access）に適した分散ファイルシステムである．データ操作の場合，共有された永続的なメモリプールに直接アクセスしてメモリコピーのオーバーヘッドを削減した．また，低遅延な RDMA を活用するためにメタデータのアーキテクチャを再設計した．

2.4 Pwrake

Pwrake[6][7] は Ruby の Rake をベースとした並列分散型のワークフローシステムである．Rake は make に似たツールであり，Makefile と同等に Ruby で記述する Rakefile が必要となる．Pwrake では Rake で実行可能なワークフローをそのまま実行することが可能であり，依存関係のない処理を分散並列実行することができる．

2.5 Montage

I/O の比重が高い天文データ処理ワークフローとして montage[7] というソフトウェアがある．montage は複数の画像を入力し一つの画像として合成（モザイク画像を生成）するための汎用ソフトウェアであり，いくつかのプログラムを組み合わせたワークフロー処理である．この montage のワークフローを Pwrake で並行分散実行した先行研究 [8] がある．

2.6 Horovod

Horovod[9] はオープンソースで開発されている Python ベースの分散学習向けフレームワークである．Horovod では Tensorflow[10], Keras[11], Pytorch[12], MXNet[13] がサポートされている．

3. 背景

バーストバッファには専用のシステムが必要になるものがある. 例えば CrayDatawarp[14] や DDN-IME[15] が挙げられる．これらのシステムは一部のシステムで導入されており, 計算ノードの数に依らず非常に高いパフォーマンスを発揮する．

一方で，バーストバッファでは計算ノード上で利用可能な NVMeSSD を用いて揮発的な分散ファイルシステムを構築することにより，確保するノード数に応じてファイルシステムの性能がスケールする．

本研究では，HPC 上で実行されうるワークロードとして MapReduce, 分散深層学習, 天文データ処理を行った．

4. 実験

4.1 概要

以下に**表 1** として実験環境を示す．**表 2** の条件のもと**表 3** について評価するために実験を行った．

表 1 実験環境

システム	Cygnus
プロセッサ	Intel Xeon Gold 6126 2.6GHz × 2CPU
スレッド数	12C / 24T × 2CPU
RAM	192GiB (DDR4-2666 ECC)
ネットワーク	InfiniBand HDR100 ×4
GPU	NVIDIA TESLA V100 (32GiB) ×4
NVMeSSD	3.2TB
PFS	DDN EXAScaler(Lustre)

表 2 実験条件

アプリケーション	データセット
Pwrake montage	2MASS images M31(7x7deg), J band
Horovod MXnet	ImageNet（ILSVRC2012） resnet50
Hadoop TeraSort	20GB

表 3 比較対象

分散バーストバッファ	BeeOND
	Gfarm/BB
PFS	Lustre

4.2 結果

次に実験結果を示す．**図 1** として pwrake によって実行した montage ワークフローの実行結果を示す．また，montage ワークフローを構成しているコマンド毎の実行時間に対するヒストグラムを，8 ノードの BeeOND を**図 2** として，8 ノードの Lustre を**図 3** として示す．なお，Gfarm/BB については有効な結果が得られなかったので省略した．

バッチサイズを 128 として 2epoch の学習を行った horovod(MXNet) の実行結果を**図 4** として示す．

Hadoop YARN(MRv2) で実行した TeraSort について，20GB の TeraGen を**図 5**，TeraSort を**図 6**，TeraValidate を**図 7** として示す．

4.3 IOR

マイクロベンチマークとして IOR[16] を用いた．IOR の POSIX インターフェイスを使用して表 2 の Read/Write に近い条件を定義し検証のための実験を行った．また，IOR では gfarm インターフェイスを有効にすることで，GfarmAPI を使用することができ，これについても実験を行った．GfarmAPI では FUSE の仕組みを用いずにファイルアクセスを行うので FUSE による性能劣化 [17] を抑制できる．

33.0MB, 5.0MiB, 410KiB のファイルに対して 1 ノードにつき 24 プロセスから Read/Write を実行した．簡単のため，今回は一回の I/O で指定したサイズ（33.0MB, 5.0MiB, 410KiB）のファイルを FilePerProcess で作成した．transferSize と blockSize は同一の値を設定し，segmentcount は

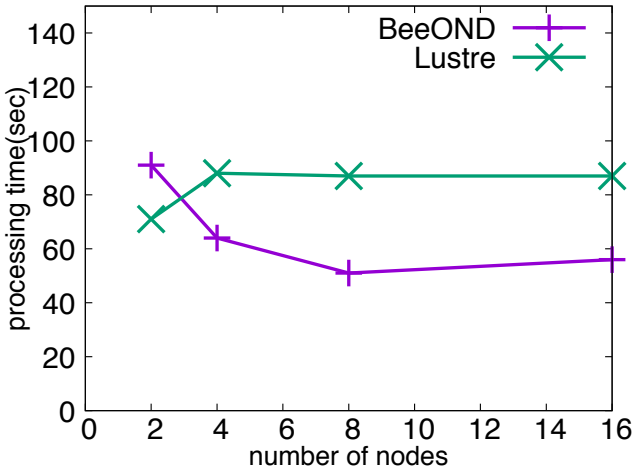

図 1　pwrake montage

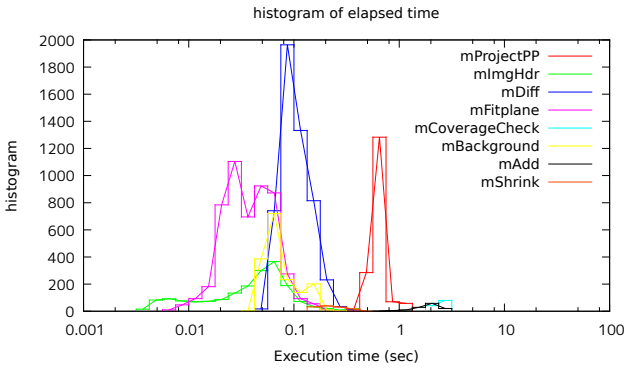

図 2　montage command histgram (8node, BeeOND)

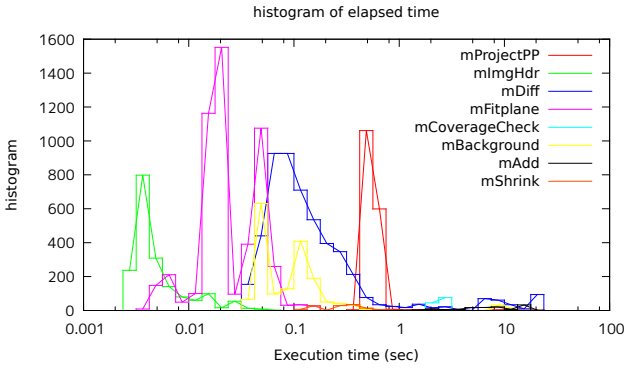

図 3　montage command histgram (8node, Lustre)

1 として，5 回試行を行った．read に関しては taskPerNodeOffset, reorderTasksConstant を有効にし，それぞれのプロセスが別ノードのプロセスから書き込まれたデータを参照するようにした．この結果を図 9，図 10 として示す．

更に，同様の条件の上で 20GiB，10GiB，40GiB の Read/Write を 3 回試行した結果について図 11 として示す．

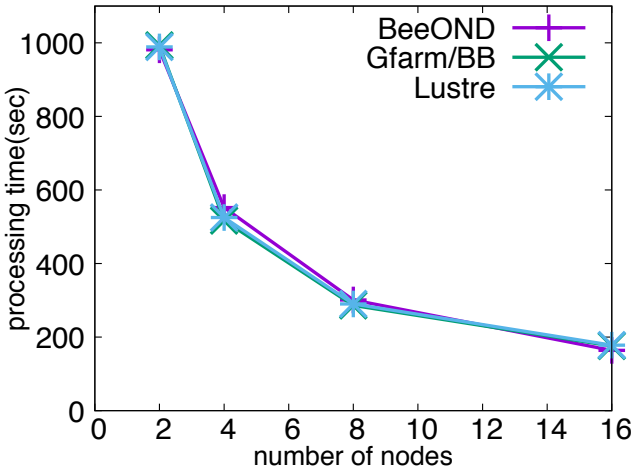

図 4　MXNet ImageNet（ILSVRC2012）2epoch training

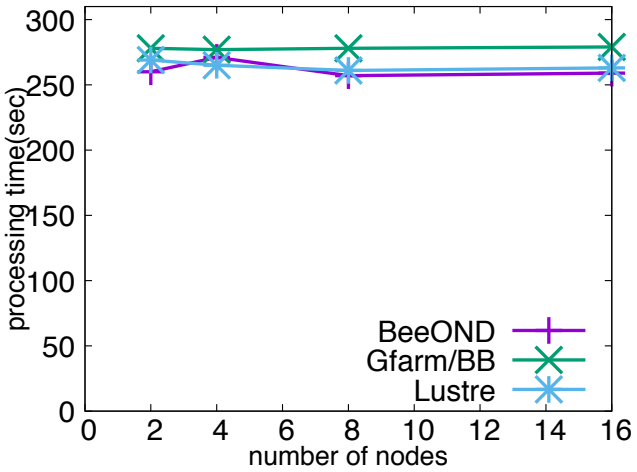

図 5　hadoop terasort teragen

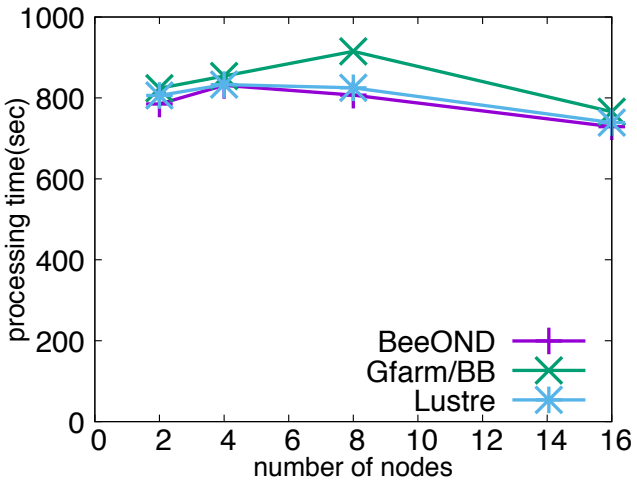

図 6　hadoop terasort terasort

5. 考察

5.1 Pwrake Montage

PwrakeMontage の結果について考える．先ず，図 2，図 3 より montage ワークフロー内で有意に多く実行されるコマ

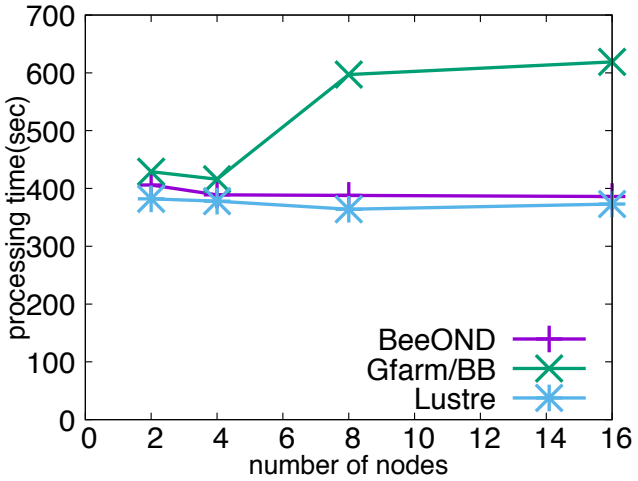

図 7　hadoop terasort teravalidate

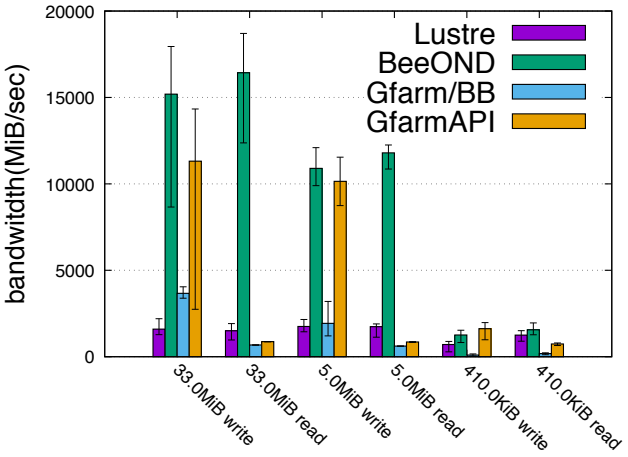

図 10　IOR 8node(24task/node) fileperproc

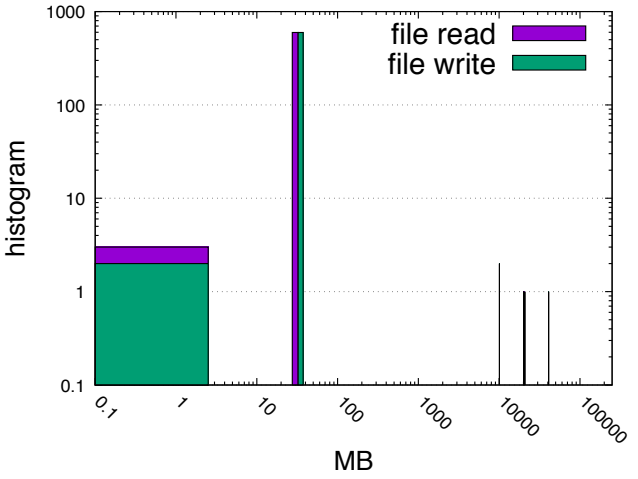

図 8　Hadoop 全体 (BeeOND 8node slaves) の実行においてファイルに発行された R/W(MB) に対するヒストグラム

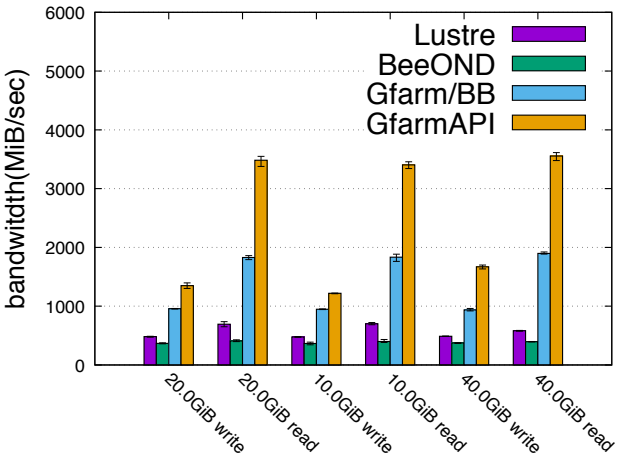

図 11　IOR 8node(1task/allnode) fileperproc

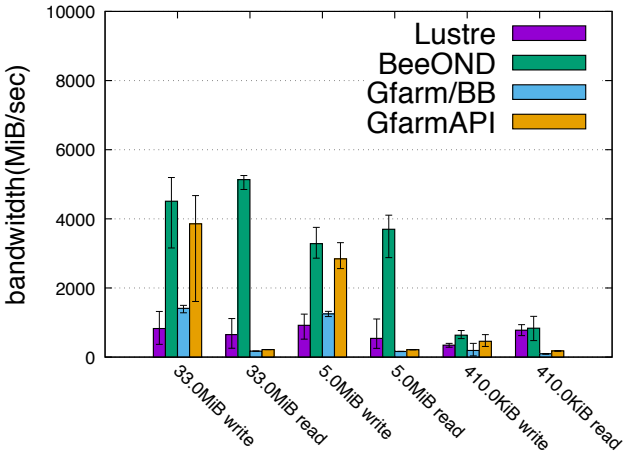

図 9　IOR 2node(24task/node) fileperproc

ンドは mProjectPP，mDiff，mFitplane であることがわかる．これらのコマンドでは，天文画像では撮影条件によって明るさにばらつきが生じるため接続する画像をなだらかにするために明るさの補正を行っている．mPrpjectPP では入力した 1 枚の元画像を処理のために変換し 1 枚の画像を生成し，mDiff では入力された 2 枚の fits イメージの重複部分の画像を生成し，mFitplane では mDiff で生成した画像を入力として画像をフィットするための補正パラメータを算出している．従って mDiff では多数の read/write が発生し，mFitplane では mDiff によって生成されたイメージに対して多数の read が発生している．mDiff では約 5MB の 2 枚の画像から 1 枚の約 410KB の差分画像を生成していた．

図 9, 図 10 より，IOR による実験結果と照らし合わせる．410KiB のファイルを読み書きする場合，BeeOND が Luster よりもやや高いパフォーマンスを出していた．特に，5 MiB の読み書きにおいて BeeOND が顕著に高い性能を出している．ここで，図 1 をみると，2 ノードでは Lustre のほうがやや実行時間が短くなっていたが 8 ノードでは実行時間の差が縮まり，16 ノードでは殆ど同じ程度の時間で実行できていた．また，IOR の実験結果では 5MiB の読み書きでは BeeOND のほうが 2 ノードであってもかなり高いパフォーマンスがあるとわかるが，図 2 より，特

に mDiff のコマンドの実行時間のばらつきが減っていることがわかる．5MiB の読み書きの性能が反映されていると考えられる．

5.2 Hadoop TeraSort

図 5, 図 6, 図 7 より，TeraSort の結果について考える．**図 8** に BeeOND にて 8 ノードのスレーブにて実行した Hadoop のワークロードにおいて，ファイルシステムのファイルへ実際に発行された I/O サイズ (MB) に対する出現頻度のヒストグラムを考える．なお，ヒストグラムの 10GB 以上について補足すると，TeraSort のある REDUCE の task で計 20GB の read の後に 40955MB の write が実行され，TeraValidate の MAP の task で計 20GB を read していた．また,TeraGen を構成している 2 つの MAP のタスクでそれぞれ 10GB の write が行われていた．

最も多く実行されているのは 33MB 程度の Read/Write ではあるが，数回だけ非常に大きなものが発行されており多くの時間を掛けていた．最も多く実行されている 33MB 程度の R/W を持つ task である，TeraSort の MAP について，このタスクは平均 2.4 秒で完了していた．図 9 より IOR 2 ノード（24task/node）の場合，BeeOND が最も良いパフォーマンスを出し，Write に関しては Lustre が Gfarm/BB よりも良い場合があった．

TeraSort のワークロードで 3 回実行される 10GB 以上のファイル I/O を伴うタスクについて考えるために，図 11 を参照する．これらのタスクは実行されたノードだけで実行されるものであり，理想的な状況としてとあるノードの 1 スレッドからのみファイルアクセスがある状況を考えた．このとき,BeeOND よりも Lustre のほうがやや高い性能を出していた．一方で Gfarm/BB は GfarmAPI が利用可能な状況であれば最も高い性能を持つことが考えられるが,POSIX でマウントされた状況であっても高い性能を出していた．しかし,TeraValidate の結果をみると Gfarm/BB で大きく実行時間が落ちていることがわかる．今回の実験では Hadoop コネクターを使用せずに LocalFS としてマウントポイントを参照し各ファイルシステムについての実験を行っているため Hadoop で実行される各タスクではファイルのローカリティが考慮されていない．特に，Gfarm/BB について，Hadoop コネクターを利用しない場合ではシングルスレッドで I/O が処理されることとなり BeeOND と比較して性能が劣化すると思われる．

5.3 Horovod MXNet

図 4 より，MXNet の結果について考える．ノード数が増加するにしたがってアプリケーションがスケールし，実行時間が短縮されていることがわかる．本実験において，Imagenet の学習時では，一枚あたり約 42KB の画像を read し，montage の mDiff では一枚あたり約 4.4MB の画像を read し約 410KB の画像を出力している．Imagenet(ILSVRC2012) では百万枚以上の画像を扱い，数千枚の画像を処理する montage と比較して小さな I/O の read が多い．図 9，図 10 より，410KiB といった細かい read が多い場合ではファイルシステム間の性能差が少ないことがわかる．また，アプリケーションの実行時間に対するファイルアクセスの割合が少ないことが考えられる．

5.4 全体

分散バーストバッファを活用してユーザーがアプリケーションを実行する場合，Lustre などの共有 PFS が持つ I/O 性能と比較してバーストな I/O 需要を満たすためには確保した計算ノードの数が重要となることが分かった．また，Lustre は共有システムのため実験を行う時間帯やタイミングによって性能のブレがあるが，バーストバッファを使えばシステムの混雑状況に関係なく性能が安定する．

分散バーストバッファによって,PwrakeMontage ではノード数が多い場合で実行時間が短縮された．HadoopTeraSort では実行時間に大きく影響することはなかったが,Hadoop コネクターを利用することでファイルのローカリティを考慮したアクセス以上に I/O 性能が改善することが考えられる．MXNet ではワークロード自体の演算が重く, 小さな I/O が多く分散バーストバッファによる恩恵を受けにくい可能性がある．

謝辞 本研究の一部は，JSPS 科研費 17H01748，JST CREST JPMJCR1414，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）および富士通研究所の助成を受けたものです．

参考文献

[1] *BeeOND: BeeGFS On Demand*. `https://www.beegfs.io/wiki/BeeOND`.
[2] Osamu Tatebe, Kohei Hiraga, and Noriyuki Soda. Gfarm grid file system. *New Generation Computing*, Vol. 28, No. 3, pp. 257–275, Jul 2010.
[3] Tatebe Osamu, Moriwake Shukuko, and Oyama Yoshihiro. Gfarm/bb - gfarm file system for node-local burst buffer. *Journal of Computer Science and Technology*, 2019.
[4] ThinkParQ. Introduction to beegfs v2.0. Technical report, June 2018. `https://www.beegfs.io/docs/whitepapers/Introduction_to_BeeGFS_by_ThinkParQ.pdf`.
[5] Youyou Lu, Jiwu Shu, Youmin Chen, and Tao Li. Octopus: an rdma-enabled distributed persistent memory file system. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pp. 773–785, Santa Clara, CA, July 2017. USENIX Association.
[6] Masahiro Tanaka and Osamu Tatebe. Pwrake: A parallel and distributed flexible workflow management tool for wide-area data intensive computing. pp. 356–359, 01 2010.
[7] Tanaka Masahiro and Tatebe Osamu. Workflow scheduling to minimize data movement using multi-constraint

graph partitioning. *Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 65–72, 2012.

[8] Tanaka Masahiro and Tatebe Osamu. Disk cache-aware task scheduling for data-intensive and many-task workflow. *Proceedings of IEEE International Symposium on Cluster Computing (Cluster)*, pp. 167–175, 2014.

[9] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*, 2018.

[10] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

[11] François Chollet, et al. Keras, 2015. `https://github.com/fchollet/keras`.

[12] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[13] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems, 2015.

[14] Cray datawarp. `https://www.cray.com/products/computing/xc-series-overview`.

[15] Ddn ime. `https://ddn.co.jp/products/ssdstorage/ime.html`.

[16] *HPC IO Benchmark Repository*. `https://github.com/hpc/ior`.

[17] Bharath Kumar Reddy Vangoor, Vasily Tarasov, and Erez Zadok. To FUSE or not to FUSE: Performance of user-space file systems. In *15th USENIX Conference on File and Storage Technologies (FAST 17)*, pp. 59–72, Santa Clara, CA, February 2017. USENIX Association.