Vol.2019-HPC-170 No.24
2019/7/25

TSUBAME3.0 におけるストレージ利用効率化のための ファイルシステムベンチマーク

野村 哲弘[1]　三浦 信一[1]　實本 英之[1]　額田 彰[1]　遠藤 敏夫[1]

概要：東京工業大学に設置されているスーパーコンピュータ TSUBAME3.0 には高速ストレージ (Lustre) およびローカル SSD を連携したジョブ毎の並列スクラッチファイルシステム (BeeOND) を備えるが，その性能特性については十分に明らかにされておらず，ユーザが適切にファイルシステムやその設定を選択するために必要な情報の提供ができていない．本報告ではこのようなファイルシステム間比較の端緒として、TSUBAME3.0 の Lustre ファイルシステムにおける並列 I/O ベンチマークの結果を報告すし，ユーザへの情報提供の可能性を探る．

1. はじめに

東京工業大学学術国際情報センター（以下,「本センター」という）では「みんなのスパコン」を合言葉とし，使いやすさと高性能を両立したスーパーコンピュータ TSUBAME シリーズを構築・運用しており，2017 年 8 月からは現行の TSUBAME3.0 [1] を供用している．2006 年に導入された TSUBAME シリーズの初代である TSUBAME1.0 以来，TSUBAME は学内外で約 1,500 名のユーザ（2018 年度ログインユーザ数）に利用されるシステムとなっている．

TSUBAME3.0 [2] の全体構成は**図 1** に示す通りであり、540 台の計算ノードと各種ストレージおよびそれらを高速に結合する OmniPath による相互結合網などからなる。

全ての計算ノードは、フルバイセクション・ファットツリートポロジで接続されており、任意の計算ノードから任意の計算ノードおよびストレージサーバにポート当たり 100Gbps のスピードで接続することができる。

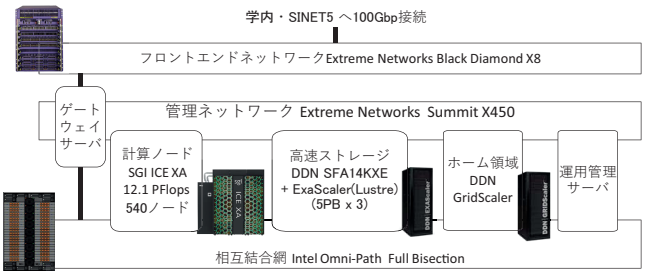


図 1　TSUBAME3.0 全体構成

ノード構成は**図 2** に示すとおりであり、1 ノードあたり 4 つの GPU、4 つの OmniPath HFI が接続されている非常に大きなノードとなっている。

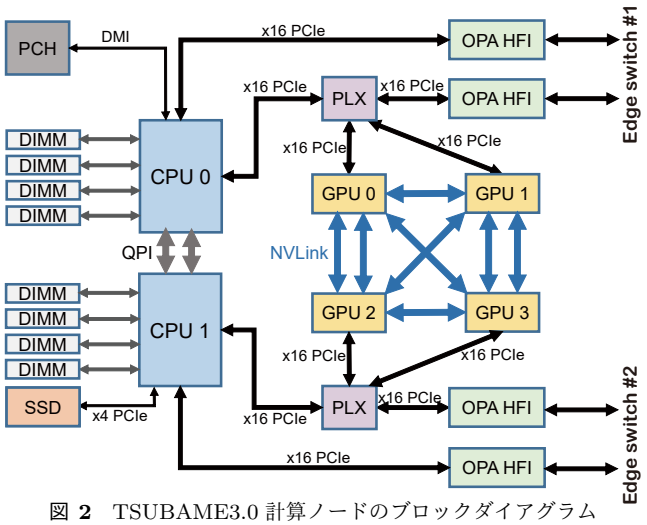


図 2　TSUBAME3.0 計算ノードのブロックダイアグラム

TSUBAME3.0 のストレージ部については，前代である TSUBAME2.0/2.5 より引き続き，全計算ノードから等しくアクセス可能なグローバルストレージ領域として，DDN SFA14KXE および EXAScaler から構成される Lustre ファイルシステムを擁している．実際に導入されているストレージは**図 3** に示すとおりである．

このような並列ファイルシステムにおいては，その性能を活かすために複数のチューニング・パラメータおよび I/O API やファイルフォーマットの選択肢が存在するが，本センターではそれらについて具体的な性能の優劣をもとにファイルシステムに合わせた選択の指針を示すこ

[1] 東京工業大学 学術国際情報センター

図 3　TSUBAME3.0 のストレージ

とがいまだ出来ていない現状がある．「京」コンピュータ（FEFS）[3] や，地球シミュレータ（ScaTeFS）[4] において，このような問題意識のもとにファイルシステムの性能データの計測と公開が行われているところであり [5]，本センターにおいても同様の計測データを公開することで，センター間におけるスパコンに関するハードウェア・ソフトウェアの機能・性能情報やユーザの利用実態と高度利用技術，運用技術に関する情報共有を加速させるとともに，利用者にとってもより良い設定でのファイルシステムの利用につながり，ひいてはファイルシステム全体としての負荷の低減にもつながると考えている．

本稿では，このような取り組みの端緒として，[5] において辻田らが計測したファイルシステムベンチマークと同様のベンチマークを実施し，FEFS および ScaTeFS との性能の差異について論じる．

2. TSUBAME3.0 高速ストレージの構成

TSUBAME3.0 においては障害範囲の局所化と運用の柔軟性の確保のため，実効容量約 5.3PB の同一構成の Lustre ファイルシステムを 3 セット備えており，本稿執筆時点ではそのうち 2 セットをユーザの利用に供している一方で，残りの 1 セットについては，将来の容量不足および Lustre ファイルシステムの機能追加時の予備として，一般ユーザの利用には供していない．

各ファイルシステムは DDN ExaScaler 1 台に SFA14KXE 1 台、SS8462 10 台のサーバで構成されており、MDS として EF4024 が 2 台、MDT として EF4024 が 2 台、ディスクとして 10TB 7.2Krpm NL-SAS HDD を 700 台搭載しているが、うち 20 台はスペアとして利用している。各 OST は 10 台のハードディスクの 8D+2P の RAID 6 として構成されており、OST の総数は 68 である。[6]

3. ファイルアクセス時のパラメータ

3.1 ファイル I/O API

Lustre ファイルシステム上のファイルへのアクセスには，POSIX 標準のシステムコールによる I/O(POSIX I/O) に加えて，Message Passing Interface(MPI) が提供する並列 I/O のための API である MPI-IO を利用することができる．

また，アプリケーションの性質やデータフォーマットに応じて各プロセスが同一ファイルの別部分を同時に I/O する方式やプロセスごとに個別のファイルを作成してアクセスする方式，MPI-IO の場合にはファイルアクセスの際にプロセス間でデータの並び替えを行うコレクティブ I/O の利用有無などが，ファイルの形状とともにアクセス性能にも強く影響する．

本測定では，下記の 4 種類の API およびファイル形式について計測を行った

- POSIX I/O
  - 単一ファイル
  - プロセス毎別ファイル
- MPI-IO
  - コレクティブ I/O なし
  - コレクティブ I/O あり

3.2 Lustre のストライピング

Lustre においては，一つのファイルを複数の OST に分散して格納するストライピングを行うことができる．ストライピングにおけるパラメータは以下のとおりである

- ストライプサイズ 分割時の各部分のデータサイズ
- ストライプカウント 分割したファイルをいくつの OST に格納するか

これらは以下のコマンドで設定することができる．

`lfs setstripe -s` サイズ `-c` カウント ファイルパス

TSUBAME3.0 におけるデフォルトはストライプサイズが 1MiB，ストライプカウントが 1 であり，1 つのファイルは単一の OST に書き込まれる設定となっている．1 つのファイルの各部分を MPI-IO の Collective I/O などで同時に読み出す場合，OST がボトルネックとなるため，ストライプカウントを増やすことが望ましい．本稿では，予備実験を行った結果良好な性能が得られる設定でああったストライプサイズ 1MiB、ストライプカウント -1(全 68OST を使用) に統一して計測を行った．

3.3 MPI-IO における Two-Phase I/O

ROMIO においては、`MPI_Info_set()` において I/O ヒント情報を与えることにより、ストレージに転送する前にデータの並び替えを行う Two-Phase I/O を行うことがで

きる。図 **4** に実際に MPI-IO に与えて、Two-Phase I/O を有効とするヒントの例を示す。

```
romio_lustre_co_ratio=16
cb_config_list=*:4
romio_cb_read=enable
romio_cb_write=enable
```

図 **4** MPI-IO ヒントの例

4. IOR によるストレージ性能計測

ストレージの基礎的な性能を計測するために，IOR 2.10.3 [7] を用いて性能計測を行った．TSUBAME3.0 の一般ユーザへの影響を最小化するため，計測には一般供用されていない Lustre 領域 hs2 を用いたが，これらのハードウェアおよびソフトウェア構成は一般供用されている hs0，hs1 の各領域と同一である．MPI は TSUBAME3.0 で標準的に使われている Open MPI 2.1.2 [8] を用いた。OmniPath のソフトウェアバージョンは 10.9 である。MPI-IO 実装は、OpenMPI に添付されている ROMIO [9] をそのまま利用している。

4.1 I/O サイズ

今回の計測における I/O のサイズを規定するパラメータを以下の通りに定義する．

- 転送サイズ (transferSize): 各プロセスが 1 度に I/O する領域のサイズ
- ブロックサイズ (BS): 各プロセスが I/O する領域の合計サイズ

8、16、36 のそれぞれのノード数において計測を行った。1 ノード当たり、GPU 搭載数と同じ 4 プロセスを起動したため、プロセス数はそれぞれ 32、64、144 となる。

4.2 IOR 計測スクリプト

これらを含めた，IOR におけるスクリプトの記載は図 **5** に示す通りである．{} で囲まれたパラメータを変化させて計測を行った．testFile は実際に書き込みを行うファイル名の接頭辞、hintsFileName は図 4 の各行に IOR が必要とする接頭辞 IOR_HINT__MPI__を加えたファイルを、Two-Phase I/O の比較を行う際のみ指定している。

```
IOR START
# common options
        reordertasksconstant=1
        fsync=1
        intraTestBarriers=1
        repetitions=5
        verbose=2
        keepFile=0
        segmentCount=1
        blockSize={BS}
        multiFile=1
        interTestDelay=1
        useO_DIRECT=1

# POSIX dedicated file tests
        filePerProc={0/1}
        api=POSIX
        testFile = {xxx}
        transferSize={transferSize}
        RUN

# MPIIO shared file tests
        fsync=0
        filePerProc=0
        api=MPIIO
        collective={0/1}
        testFile = {xxx}
        transferSize={transferSize}
        hintsFileName= {xxx}
        RUN

IOR STOP
```

図 **5** IOR ベンチマーク計測スクリプト

5. 性能計測結果と考察

IOR による各条件における計測結果を図 **6**〜図 **13** に示す。ベースラインとなる POSIX I/O 性能は特に File Per Process の場合において高い性能を示しており、アプリケーション側の制約がない限りにおいては、単一ファイルにまとめようとせずにプロセスごとに別ファイルに書き出す方が読み書きともに効率が良いことがわかる。

また、読み込み性能において、36 ノード、BS=2048MiB の場合にのみ顕著な性能低下が観測された。これは他の計測において OST および各計算ノードのファイルキャッシュが顕著に効いていることを示唆している。

上記の 1 例を除いて、BS および transferSize による性能への影響はあまりなく、転送のサイズや転送単位によるスループットへの影響は軽微であるといえる。

MPI-IO については、Collective I/O API の使用の有無によって性能が大きく左右され、Collective I/O を用いない場合、POSIX I/O を用いた場合と比べて顕著に良い Write 性能が得られていることがわかる。

また、Collective I/O を使う場合において、特定の transferSize において Read 性能が顕著に低下していることがわかる。これは、Two-Phase I/O の自動選択によるものと考えられる。図 **14** に、同様の実験を Two-Phase I/O を明示的に有効および無効にした際の性能を示す。TSUBAME3.0 の Lustre 領域に対しては Two-Phase I/O を一律で無効化したほうが良いアクセス性能が得られる。

6. おわりに

TSUBAME3.0 の高速ストレージにおけるファイルシステム性能の調査の一環として、Lustre ファイルシステム

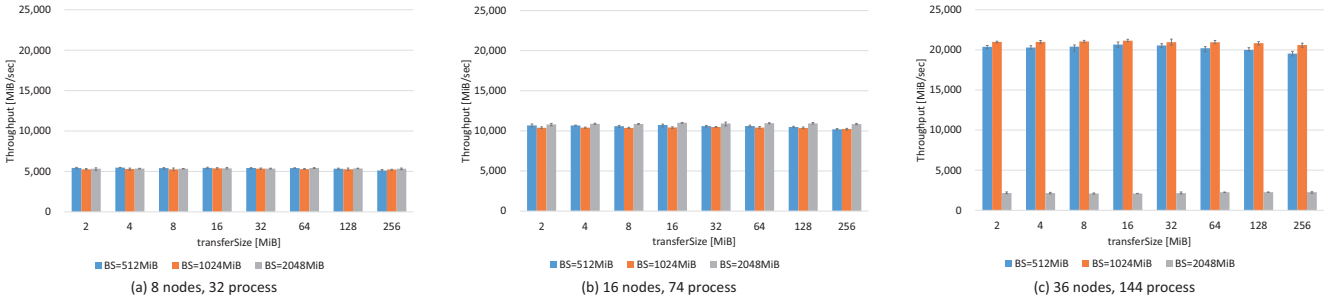


図 6 POSIX Read (Single Shared File)

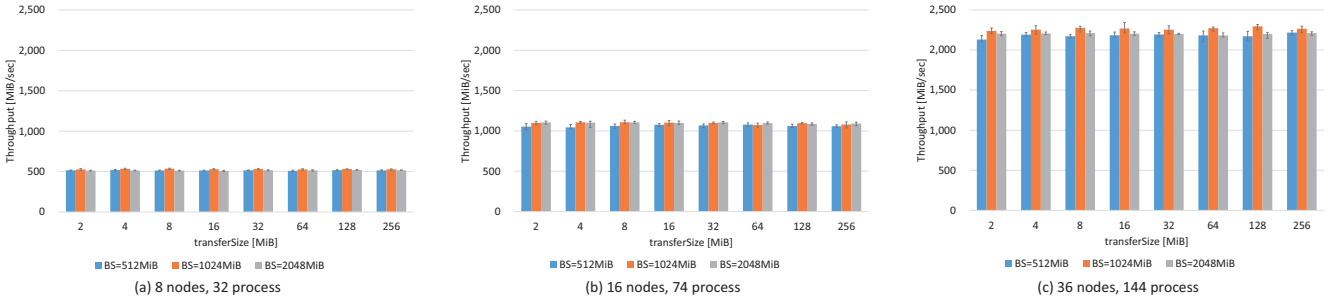


図 7 POSIX Write (Single Shared File)

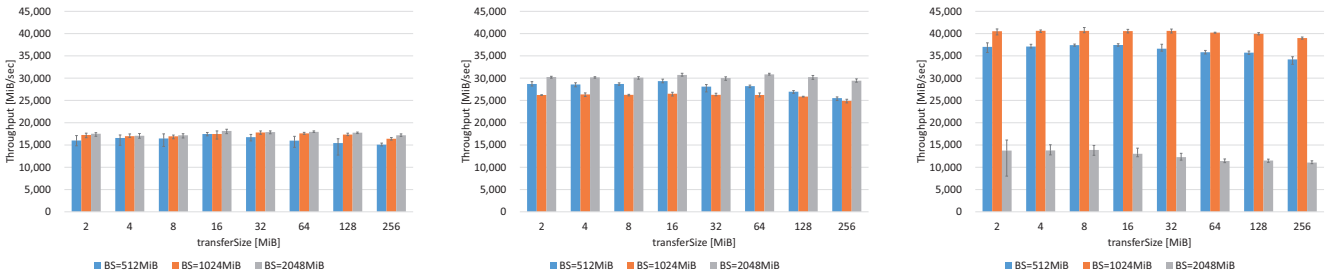
の基礎的なベンチマークを行った。ベンチマークの探索空間が広く、ストライプサイズの変更や、ストライプカウントの変更については、今回は予備実験で最良の結果が出た 1 パラメータに固定して行ってしまっているが、こちらについても網羅的に計測を行って他の機関のスーパーコンピュータと比較できるようにする必要がある。また、TSUBAME3.0 の計算ノードには NVMe SSD が搭載されており、これらを組み合わせてジョブ内からのみ使われる局所化した並列ファイルシステムである BeeGFS [10] を形成できる BeeOND [11] が導入されており、グローバルファイルシステムであるところの Lustre との性能の比較が必要である。今後は、上記ベンチマークの網羅性の改善に加えて、アプリケーションに合ったファイルシステムやその設定を提示できるよう、データの羅列にとどまらずユーザへの指針も含めて公開できるようにしたい。

謝辞 TSUBAME3.0 の設計には，東京工業大学学術国際情報センターが推進してきた文部科学省「スパコン・クラウド情報基盤におけるウルトラグリーン化技術」および「スマートコミュニティ実現のためのスパコン・クラウド情報基盤のエネルギー最適化の研究推進」，JST CREST(JPMJCR1303, JPMJCR1501) などのプロジェクトの研究成果が活用されている．

また，今回の計測にあたり，[5] の著者である JAMSTEC および理研 R-CCS の皆様，TSUBAME3.0 のストレージベンダーである DDN の皆様に多大な助言をいただきました．ここに感謝いたします．

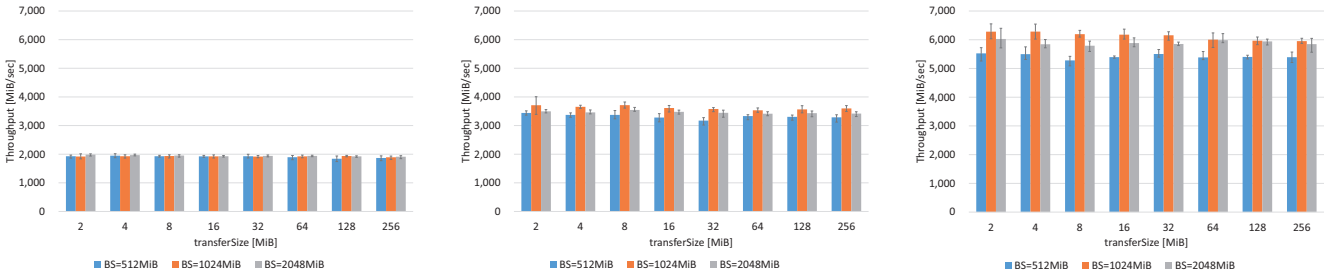
参考文献

[1] 東京工業大学学術国際情報センター：TSUBAE 計算サービス，https://www.t3.gsic.titech.ac.jp/.
[2] 松岡聡，遠藤敏夫， 額田彰，三浦信一，野村哲弘， 佐藤仁，實本英之，Drozd, A.：HPC とビッグデータ・AI を融合するグリーン・クラウドスパコン TSUBAME3.0 の概要，情報処理学会研究報告，Vol. 2017-HPC-160, No. 29, pp. 1–6 (2017).
[3] 理化学研究所計算科学研究センター：「京」について，https://www.r-ccs.riken.jp/jp/k/.
[4] 国立研究開発法人海洋研究開発機構：地球シミュレータ，http://www.jamstec.go.jp/es/jp/.
[5] 辻田祐一，中川剛史，板倉憲一，宇野篤也：ファイルシステムの利用高度化に向けたスパコンセンター間での情報共有の取り組み，情報処理学会研究報告，Vol. 2018-HPC-165, No. 6, pp. 1–10 (2018).
[6] : TSUBAME 講習会資料: Lustre Seminar，https://www.t3.gsic.titech.ac.jp/lectures.
[7] Laboratory, L. L. N.: High-Performance Computing, https://github.com/hpc/ior.
[8] : Open MPI: Open Source High Performance Computing, https://www.open-mpi.org/.
[9] Thakur, R., Gropp, W. and Lusk, E.: On Implementing MPI-IO Portably and with High Performance, *Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems*, IOPADS '99, New York, NY, USA, ACM, pp. 23–32 (online), DOI: 10.1145/301816.301826 (1999).
[10] Fraunhofer Center: BeeGFS - The Leading Parallel Cluster File System, https://www.beegfs.io/.
[11] Fraunhofer Center: BeeOND: BeeGFS On Demand, https://www.beegfs.io/wiki/BeeOND.



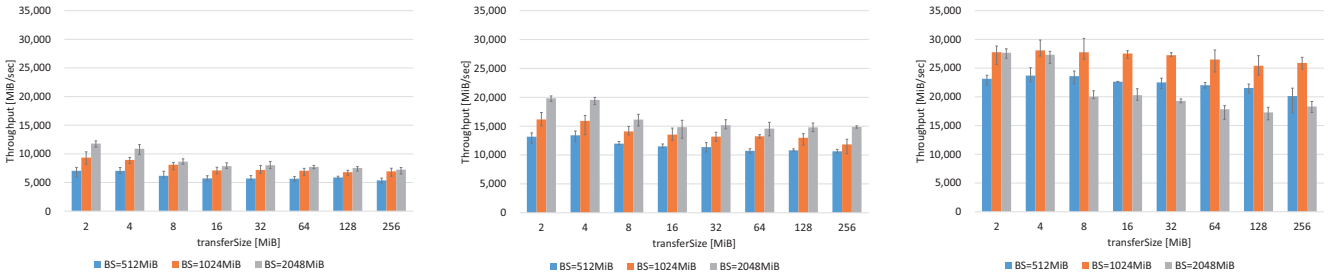
(a) 8 nodes, 32 process (b) 16 nodes, 74 process (c) 36 nodes, 144 process

図 8 POSIX Read (File Per Process)



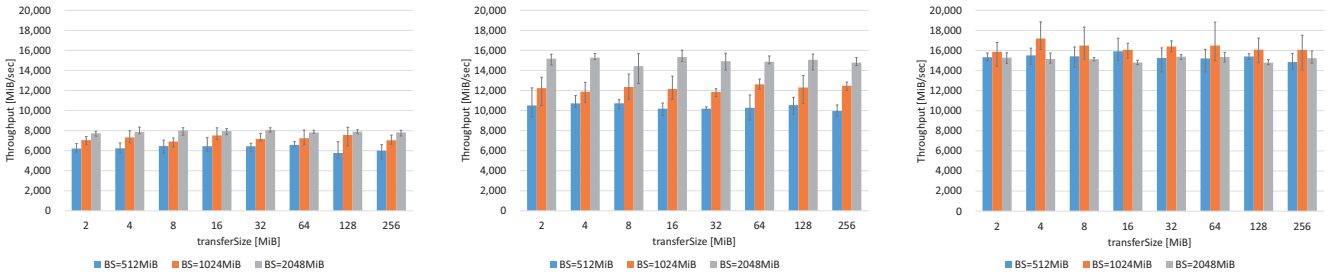
(a) 8 nodes, 32 process (b) 16 nodes, 74 process (c) 36 nodes, 144 process

図 9 POSIX Write (File Per Process)



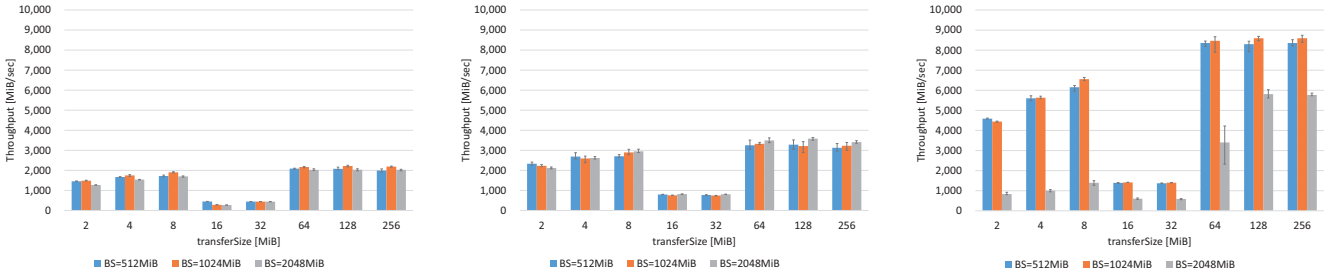
(a) 8 nodes, 32 process (b) 16 nodes, 74 process (c) 36 nodes, 144 process

図 10 MPIIO Read (Collective I/O なし)



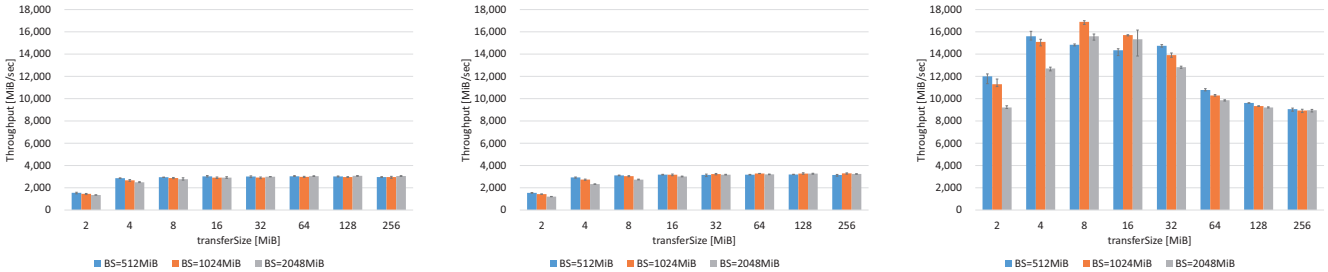
(a) 8 nodes, 32 process (b) 16 nodes, 74 process (c) 36 nodes, 144 process

図 11 MPIIO Write (Collective I/O なし)



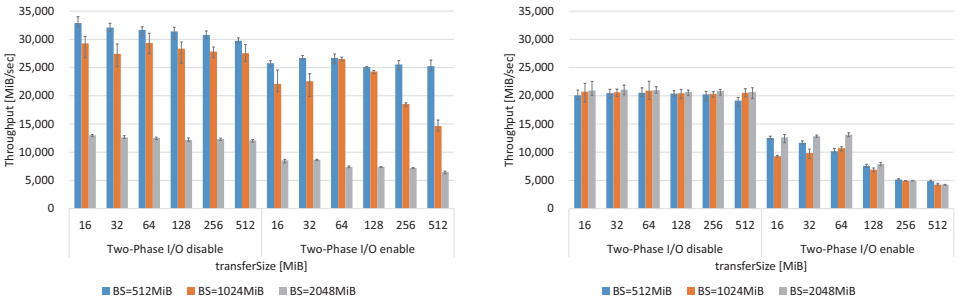
(a) 8 nodes, 32 process (b) 16 nodes, 74 process (c) 36 nodes, 144 process

図 12 MPIIO Read (Collective I/O あり)



(a) 8 nodes, 32 process (b) 16 nodes, 74 process (c) 36 nodes, 144 process

図 13 MPIIO Write (Collective I/O あり)



Read Write

図 14 MPIIO Two-Phase I/O の有無による性能比較 (72nodes, 288process)