Vol.2011-HPC-130 No.40
2011/7/28

大規模並列ファイルシステムに対するワークフローアプリケーションのI/O性能解析

斎藤貴文[†1]　佐藤仁[†1]　松岡聡[†1,†2,†3]

大規模データを扱うデータインテンシブアプリケーションの増加とスーパーコンピュータに代表される大規模並列分散環境の増加に伴い並列ファイルシステムが注目を集めている。並列ファイルシステムは高性能 I/O を実現するために用いられるが、実際のワークフローアプリケーションのデータのようにデータサイズの大量のデータサイズの小さいファイルに対する I/O やメタデータアクセスの解析は未だ不十分である。また我々は先の研究においてデータインテンシブワークフローを並列ファイルシステム上において実行したところ、I/O 性能に影響を及ぼす要因として計算ノードのメモリにおけるバッファキャッシュと並列ファイルシステムにおけるメタデータアクセスを確認した。本稿ではこれら性能に影響を及ぼす要因について TSUBAME2.0 上で実ワークフローアプリケーションを動かすことで実稼働している二つの並列ファイルシステム (Lustre, GPFS) の比較を行った。その結果、バッファキャッシュは両方のファイルシステムで確認されたが、GPFS よりも Lustre のほうが明確に現れたことが観察された。またメタデータアクセスについてもどちらのファイルシステムにおいてもメタデータサーバへの競合が確認され、メタデータサーバの構成によって影響の度合いが異なることが確認された。

Analysis of Workflow Aplication I/O Performonce on Large Parallel File System

Takafumi Saito ,[†1] Hitoshi Sato [†1]
and Satoshi Matsuoka[†1,†3]

Data-intensive scientific applications are becoming popular in astronomy and high-energy physics and parallel filesystems are attracting attentions to acquire tremendous high I/O throughput on a parallel distributed large-scale environment such as supercomputers. However, performance surveys on file-grained I/O and metadata access of workflow applications on parallel file systems are not investigated. This paper reports the performance comparison of fine-grained I/O on Lustre and GPFS file systems in the TSUBAME2.0 supercomputer by using a workflow application called Montage. We observed buffer cache effects of a running application on both file systems; however, the effect clearly became visible on Lustre than on GPFS. Besides, we also observed metadata contentions on both file systems; the influence of the contentions showed different characteristics depending on the composition of the metadata management.

1. はじめに

近年、様々なバイオインフォマティクス、天文学などの科学分野において大量のデータに対する解析処理への需要が高まっている。それに伴い大規模データに対する並列分散処理を多数のノードにスケールアウトさせ高速にデータ処理を行うデータインテンシブアプリケーションが発達している。代表的なものとして MapReduce[1] が挙げられるが、既存のプログラムの集合体で容易に実行でき多くのプラットフォームへの移植性が高いワークフローアプリケーションも注目を集めている。[2] このようなワークフローアプリケーションの実行環境としてスーパーコンピュータ (スパコン) などに代表される大規模並列分散環境が利用されているが、このような環境においてはファイルの I/O 性能向上のために並列ファイルシステムが広く用いられている。

既存の並列ファイルシステムでは I/O スループットの向上を重視しているが、実際のワークフローアプリケーションでは大量のサイズの小さなファイルにアクセスすることが多く、それに伴ってメタデータアクセスの頻度も高い。このため、単に I/O スループットの性能向上だけでなく、細粒度の I/O においてのメタデータアクセス性能の向上がこれからの大規模データ処理においては重要である。我々は先の研究[3] でワークフローアプリケーションの性能モデルを構築し、Lustre ファイルシステム上でのワークフローアプリケーションの I/O 性能を考察した。その結果、ワークフローアプリケーションの I/O 性能では、主に、(1) 計算ノードにおけるメモリのバッファキャッシュの影響、(2) 並列ファイルシステムにおいてのメタデータアクセス性能、の要因があることが明らかになった。しかし、その調査は不十分で、厳密な I/O 性能の解析は出来ていない。本稿では、プロダクティブに運用されている TSUBAME2.0[4] 上の二つの並列ファイルシステム (Lustre, GPFS) に対し、実際のワークフローアプリケーションを実行し、我々が先の研究で確認したこれらの I/O 性

†1 東京工業大学
†2 国立情報学研究所
†3 科学技術振興機構

能の要因を調査した。その結果ワークフローアプリケーションの I/O に影響を及ぼす要因が確認された。計算ノード上のメモリのバッファキャッシュは両方のファイルシステムにおいて観測できたが、明確にスループットの差が現れた Lustre に対し GPFS では大きくスループットに差が開くことはなかった。また並列ファイルシステムのメタデータアクセスについてはプロセス数が増加するとメタデータの競合が観測でき、単一のメタデータサーバにアクセスが集中した Lustre とメタノードで負荷分散がなされた GPFS で対照的なメタデータの挙動が確認できた。

2. 並列ファイルシステム

並列ファイルシステムとは複数のディスクを単一のストレージとしてみせるファイルシステムである。複数のディスクに I/O を行うことで負荷分散がなされ、ファイルシステムの信頼性を向上させる。またファイルを分割し複数のディスクに分散させることで高バンド幅を確保し I/O 性能の向上を図る。この手法をストライピングという。

2.1 Lustre

Lustre[5] とはオープンソースの大規模並列ファイルシステムである。Lustre のコンポーネントはファイルシステムをマウントする Lustre クライアント、メタデータを管理する MDS(Meta Data Server) とそれを保管する MDT(Meta Data Target)、ファイルへの I/O リクエストを扱う OSS(Object Storage Server) そしてファイルデータを格納する OST(Object Storage Server) である。ファイルを参照するためにはまずクライアントが MDS からファイルのメタデータを獲得し、メタデータをもとにデータオブジェクトが保存されている OSS にインタラクティブにアクセスする。Lustre のファイルデータは設定されたチャンクサイズに従ってストライピングを施され、各オブジェクトは複数のディスクに分散して保管される。また、Lustre ではストライピングのチャンクのサイズや数を設定することができる。TSUBAME2.0 における Lustre ファイルシステムの一つは MDS が 2 台、OSS が 4 台で構成されている。MDS のうち 1 つはフェイルオーバーの為に待機状態であり、2 つの MDS に対して MDT1 つが接続されている。また OSS2 台に対して 28 個の OST が接続されファイルシステム全体では 56 台の OST が存在する。図 1 は TSUBAME2.0 における GPFS の構成を表している。

2.2 GPFS

GPFS(General Parallel File System)[6] は IBM が開発したクラスタ向けのファイルシステムである。GPFS も Lustre と同様にファイルにストライピングを行い複数ディスクに分散させるが、GPFS はファイルを等分割しすべてのディスクに分散して配置する。また GPFS では各ファイルに対し一つのファイルサーバが「メタノード」として該当ファイルのメタデータの管理を行うため、厳密なメタデータサーバは存在しない。TSUBAME2.0 において GPFS ファイルシステムはディスクが 56 台で構成されおり、そのうちファイルシステムのストレージとして扱うのは 48 台である。図 2 は TSUBAME2.0 における GPFS の構成を表している。

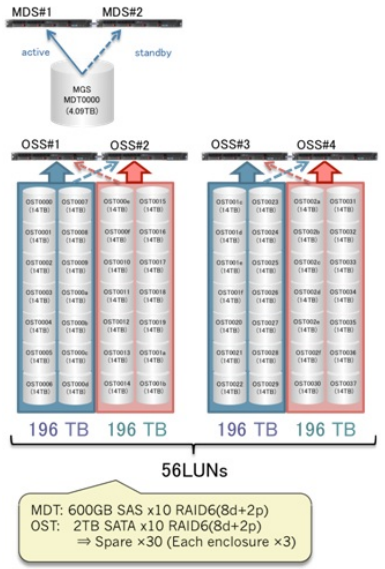

図 1 Lustre の構成
Fig. 1 configuration of Lustre

3. ワークフローアプリケーションの I/O

3.1 ワークフローアプリケーション

ワークフローアプリケーションとは複数のタスクから構成され、タスクの依存関係からアプリケーションの処理が定義される。タスクの処理によって並列に処理を行うことができる。ここではタスクの処理を行うプロセスをジョブとよぶ。ジョブ間の通信には中間ファイルを用いる。あるジョブの出力ファイルとして書き出された中間ファイルは次の処理を行

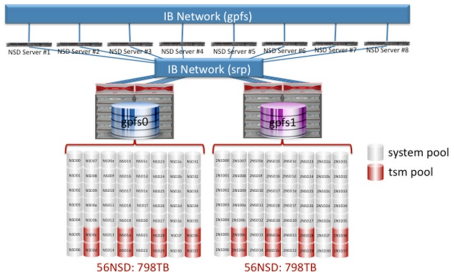

図 2 GPFS の構成
Fig. 2 configuration of GPFS

うジョブの入力ファイルとして読み出される。ワークフローアプリケーションでは並列ファイルシステム上に中間ファイルを書きこむため MPI のように明示的にデータ転送を行う必要がなくアプリケーションの利用者の煩雑な処理を隠蔽することが可能である。タスクによっては並列に処理可能な部分と逐次でしか処理できない部分に分けられる。ジョブをノード、中間ファイルをエッジとすることでワークフローアプリケーションを無閉路有向グラフ（DAG）で表すことが可能である。

3.2 スパコンにおけるワークフローアプリケーションの I/O

先の研究[3] において我々はワークフローアプリケーションの性能モデルを構築し、この性能モデルから分散環境におけるワークフローアプリケーションの実行時間の予測を行った。しかし、予測と実測と比較したところ二つの実行時間には大きな差異が生じた。ワークフローアプリケーションの I/O の考察を行った結果、I/O に影響を及ぼす要因が明らかになった。

3.2.1 計算ノード上のメモリのバッファキャッシュの影響

ワークフローアプリケーションをノード数を増やして実行したところノード数が増加するに従って Read のスループットが低いジョブも増加する現象が見受けられた。このような現象が生じる原因はバッファキャッシュであると考えられる。ワークフローアプリケーションでは各ジョブはファイルに出力結果を書きだすので、そのデータはジョブを実行したノードのメモリにキャッシュとして乗るはずである。もしその出力されたファイルを入力とするジョブが同じノードで実行されたならば、その入力ファイルのデータはキャッシュに乗っているので高スループットでファイルを読み出せるはずである。一方次の処理を行うジョブが別のノードで実行された場合は、ストレージサーバのディスクからデータを読み出さなければいけないので低スループットとなる。よってノード数が増えるにつれ中間ファイルが各ノードのメモリに点在し、キャッシュのヒット率が低下したため低スループットのジョブが増加したと推測できる。

3.2.2 並列ファイルシステムにおけるメタデータアクセス性能

ワークフローアプリケーションのジョブの行う処理は入力ファイルを読み込む Read 時間、タスクによって定められた処理を行う計算時間、計算結果を出力ファイルに書きだす Write 時間の三つに分割することができる。性能モデルではこれら三つの処理の合計をジョブの処理時間と定めた。しかし、実際の実行時間と性能モデルから予測した実行時間を比べると後者が前者を下回る結果となった。この原因として性能モデルにおけるメタデータへのアクセス時間の無視が考えられる。ワークフローアプリケーションの実行時のメタデータアクセスの実行時間を積算で求めたところ三つの処理時間と同様に実行時間のうちで大きな割合を占めていることが判明した。このことからワークフローアプリケーションにおいてメタデータアクセスというのは性能に影響を及ぼす重大な要素だということが推測された。

4. 実　　験

実験では TSUBAME2.0 の二つのファイルシステムの I/O とメタデータアクセスの性能を調査する。まず IOzone[7] というベンチマークを用いて Lustre と GPFS それぞれの I/O スループットを計測した。次にワークフローアプリケーションの一つである Montage を両ファイルシステム上で実行し実アプリケーションの I/O とメタデータアクセスの挙動を観測した。

Lustre のストライピングのチャンクサイズは利用者が設定することができるが、今回の二つの実験では TSUBAME2.0 のデフォルト値である 1MB でおこなった。

4.1 ファイルシステム基本性能の比較

4.1.1 実 験 環 境

I/O 性能を計測するベンチマークである IOzone[7] を用いてベンチマーク二つのファイルシステムの基本的な I/O 性能の比較を行った。IOzone で-t オプションでプロセス数を指定すると、IOzone は並列実行を行う。IOZone を並列に実行すると各プロセスは対応するファイルにのみ I/O を行う。この論文における実験では指定したノードで分散並列実行するクラスターモードで iozone を複数ノードで並列に実行した。実験環境は TSUBAME2.0 の S キューを用いた。S キュー 1 ノードのコア数は 12 であるが、ハイパースレッディングによって利用者からは仮想的に 24 コアにみえる。また S キューのメモリ容量は 54GB である。

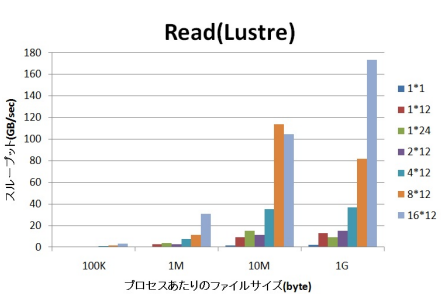

図 3 IOzone における Read の挙動 (Lustre)
Fig. 3 behavior of read operation by IOzone (Lustre)

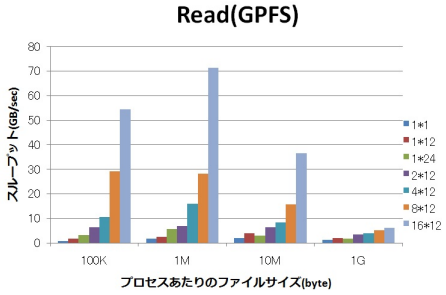

図 4 IOzone における Read の挙動 (GPFS)
Fig. 4 behavior of read operation by IOzone (GPFS)

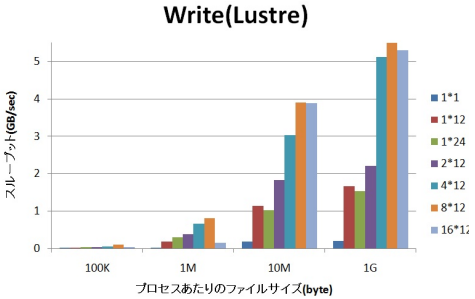

図 5 IOzone における Write の挙動 (Lustre)
Fig. 5 behavior of write operation by IOzone (Lustre)

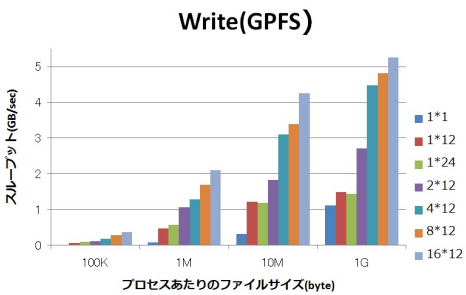

図 6 IOzone における Write の挙動 (GPFS)
Fig. 6 behavior of write operation by IOzone(GPFS)

4.1.2 実験結果

図 3-4 は IOzone を複数プロセスでファイルサイズを変えて実行したとき、それぞれの全プロセスの Read のスループットの合計を表している。図 3-4 より、二つのファイルシステムのを比較するとファイルサイズが小さいときは GPFS が Lustre より高いスループットを出している一方でファイルサイズが大きくなるに従って Lustre の方がスループットが高い。図 5-6 は Write のスループットの合計を表している。図 5-6 を比較するとファイルサイズが 1MB のとき GPFS のスループットが Lustre の約二倍であることがわかる。これは Lustre ではストライピングのチャンクサイズが 1MB であるのでファイルがストライピングされないのに対し、GPFS はすべてのディスクにデータオブジェクトを保存するようにストライピングを施すので複数ディスクにデータが分散している GPFS のスループットの方が高い値になったと考えられる。ファイルサイズが 10MB、1GB になると GPFS と Lustre のスループットの挙動は似たような傾向になっていることがわかる。この時 Lustre もファイルのストライピングを行っているため、両ファイルシステムにおける I/O の負荷が分散しているためと推測できる。

4.2 ワークフローアプリケーションにおける I/O 性能評価

4.2.1 実験環境

ワークフローアプリケーションである Montage を実行し、実際のワークフローアプリケーションの I/O 性能を調べた。Montage は複数の天体画像を組み合わせて最終的に一つの画像にまとめるアプリケーションである。Montage の特徴として入力・中間ファイルのサイズが小さく、殆どのファイルが高々数 MB であることが挙げられる。またファイルへの追記は行わない。図 7 は Montage を DAG で表したものである。

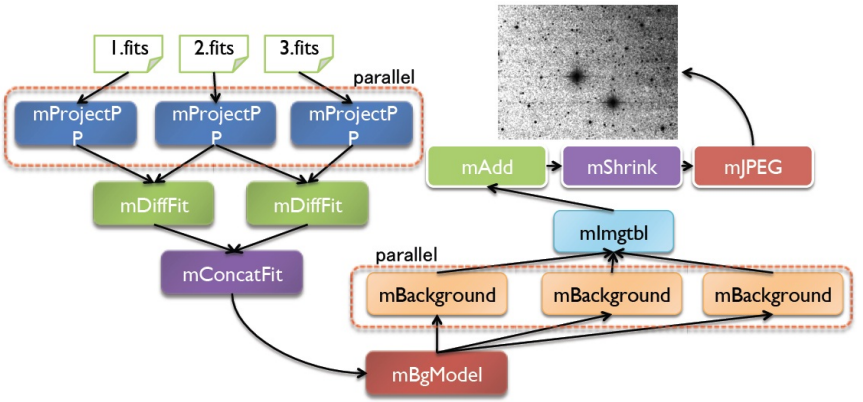

図 7 Montage の DAG
Fig. 7 DAG of Montage warkflow

Montage を実行するためのミドルウェアとして GXP Make を用いた。GXP[2] は分散環境で並列処理を行うためのツールである。マシンのバックグラウンドでデーモンが協調して

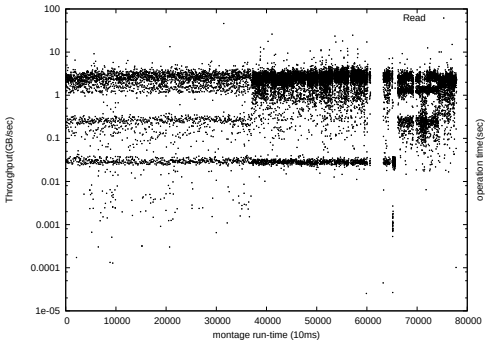

図 8 Read のスループットの挙動 (1 ノード 1 プロセス, Lustre)
Fig. 8 behavior of read throughput (1 node, 1 prosess, Lustre)

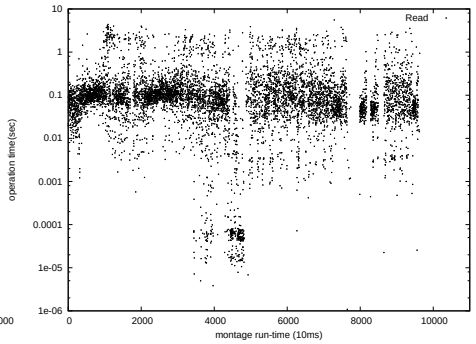

図 9 Read のスループットの挙動 (32 ノード 24 プロセス, Lustre)
Fig. 9 behavior of read throughput (32 nodes, 24 prosesses, Lustre)

命令を実行する。GXP Make[⋆1]は Makefile に記述されている処理を GXP を用いて分散環境で実行する。また、Montage の実行のログを獲得するために logfuse [⋆2]というツールを用いた。logfuse は FUSE でマウントしたディレクトリ下で I/O やメタデータの生じた時間や Read/Write したデータサイズをログに掃き出す。

Montage は S96 キューで実行した。S96 キューは S キュー同様 12 コア (仮想的には 24 コア) であるが、メモリ容量が 96GB となっている。今回の実験では GXP Make を用いて Montage を S96 キュー上で並列実行した。Montage は FUSE でマウントしたディレクトリ下で実行され、logfuse によってログを獲得する。この実験で用いた Montage のデータセットは入力ファイルのサイズの合計サイズが 1.2GB、中間ファイルの合計サイズが 7.5GB、総ジョブ数が 1542 であるものを用いた。

4.2.2 実験結果

まず Montage における Read/Write 性能を計測した。図 8-9 は Lustre で Montage の実行した際に生じた Read のスループットを 10 ミリ秒間隔で計測したグラフである。図 8 は 1 ノード 1 プロセスで、図 9 は 32 ノード 24 プロセスで Montage を実行している。図 8 と図 9 を比べると、図 8 ではスループットが 1GB/sec 付近に半分以上の点が集まってい

⋆1 http://www.logos.ic.i.u-tokyo.ac.jp/gxp/
⋆2 http://www.intriger.jp/wiki/index.php/Profiler_for_GXP_make

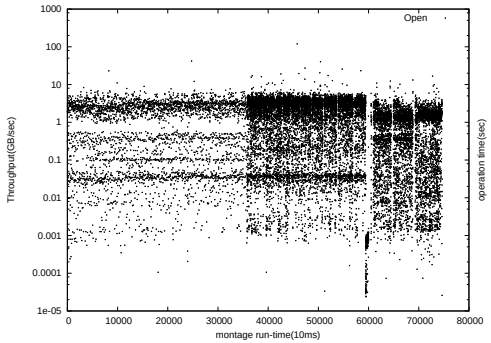

図 10 Read のスループットの挙動 (1 ノード 1 プロセス,GPFS)
Fig. 10 behavior of read throughput (1 node, 1 prosess, GPFS)

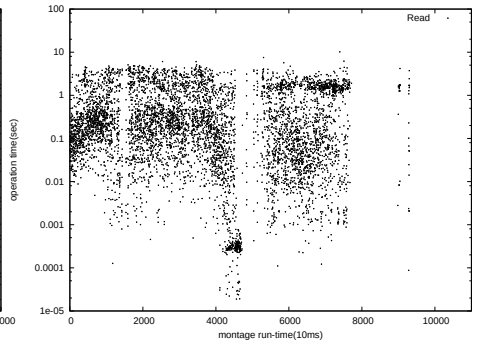

図 11 Read のスループットの挙動 (32 ノード 24 プロセス,GPFS)
Fig. 11 behavior of read throughput (32 nodes, 24 prosesses, GPFS)

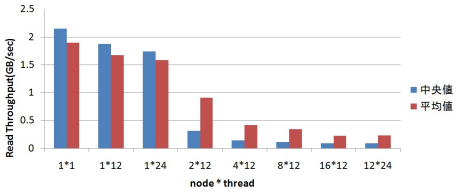

図 12 Read スループットの平均値と中央値 GPFS
Fig. 12 average value and median value of read throurhput (Lustre)

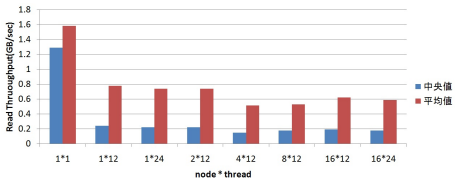

図 13 Read スループットの平均値と中央値 (GPFS)
Fig. 13 average value and median value of read throurhput (GPFS)

るのに対し、図 9 では点が 100MB/sec 付近に密集していることがわかる。このスループットの差異はバッファキャッシュによって引き起こされていると考えられる。図 8 をみると 40000ms 以降、1GB/sec 以上のスループットが増加しているのがわかる。40000ms の時点で Montage は最初のタスクが終了し次のタスクの実行を開始している。最初のタスクで出力したファイルはすべてノードのメモリに乗っており、次のタスクはそのファイルを入力ファイルとして読み込む時にメモリのキャッシュにヒットしスループットが向上している。ノード数が増加すると中間ファイルを出力するジョブと中間ファイルを入力とする次のタスクのジョブが同じノードで実行する頻度が現象するためバッファキャッシュの効果がなくスループットが低下する。図 10 と図 11 は GPFS における Read のスループットを表している。GPFS においても Lustre と同様に 1 ノードのときは 1GB/sec で付近に点が集中する。(図 10-11) しかし図 9 より GPFS は Lustre と異なり 32 ノードのとき点が広い範囲のスループットの間に分散している。

図 12 と図 13 は 10ms 間隔で計測した Montage の実行時に生じた Read のスループットの平均値と中央値を表したグラフである。また図 12 より Lustre において 1 ノードのとき Read のスループットの平均値と中央値はいづれも 1.5GB/sec 以上に到達しており、ノード数が増えるに従ってスループットの平均値と中央値が減少している。このようにバッファキャッシュの影響がみてとれる。一方 GPFS では 1 ノード 1 プロセスでは高スループットを出しているに関わらず、1 ノード 12 プロセスのときは中央値と平均値どちらも複数ノードで実行したときとほぼ変わらない値となっている。この結果と図 11 から、GPFS で実行したジョブの Read のスループットはばらつきが大きくバッファキャッシュの影響が Lustre ほど顕著に現れなかった。また、両ファイルシステムの複数ノードにおけるスループットの平均値を比べると Lustre よりも GPFS の方が高いスループットが出ていることがわかり、GPFS の方が並列アクセスの性能が高いと推測できる。

また Montage ではファイルアクセス時に生じるメタデータアクセスの実行時間も計測した。図 14-14 は Montage が実行中に行った getattr 命令の処理時間の 10ms 間隔の平均値を表したグラフである。図 14 が 1 ノード 1 プロセスで、図 14 では 32 ノード 24 プロセスで、どちらも Lustre を用いて実行した。図 14-14 から 1 ノード 1 プロセスのときは多くの点が 1e-05 秒に集まっているのに対し、32 ノード 24 プロセスではほとんどの点が 1e-05 秒以上の処理時間で点在している。これよりプロセス数が増えると getattr 命令の実行時間は増加すると考えられる。図 18 と図 19 はノード数とプロセス数を変えて Montage 実行した際に生じた getattr 命令の処理時間の中央値のグラフである。図 18 と図 19 からもプロセ

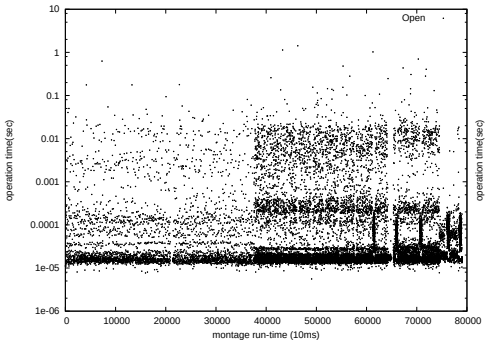

図 14 getattr 命令の挙動 (1 ノード 1 プロセス, Lustre)
Fig. 14 behavior of getattr operation (1 node 1 prosess, Lustre)

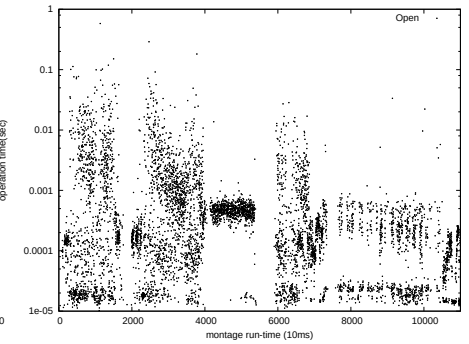

図 15 getattr 命令の挙動 (32 ノード 24 プロセス, Lustre)
Fig. 15 behavior of getattr operation (32 nodes, 24 prosesses, Lustre)

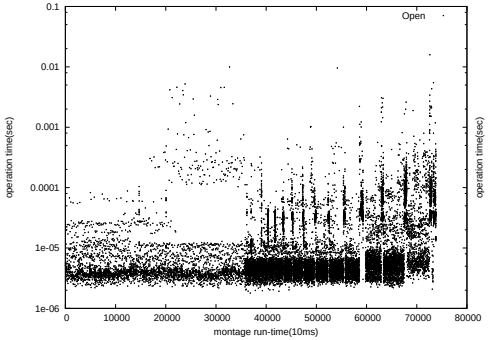

図 16 getattr 命令の挙動 (1 ノード 1 プロセス, GPFS)
Fig. 16 behavior of getattr operation (1 node, 1 prosess, GPFS)

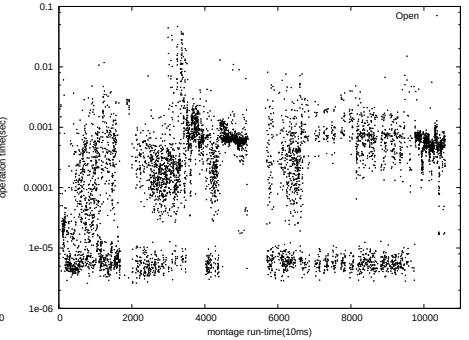

図 17 getattr 命令の挙動 (32 ノード 24 プロセス, GPFS)
Fig. 17 behavior of getattr operation (32 node, 24 prosesses, GPFS)

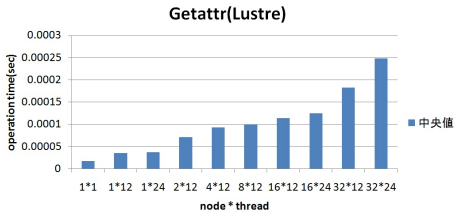

図 18 getattr 命令の処理時間の平均値と中央値 (Lustre)
Fig. 18 average value and median value of getattr operation run-time(Lustre)

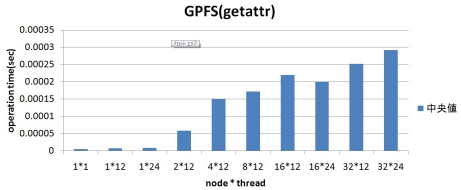

図 19 getattr 命令の処理時間と平均値と中央値 (GPFS)
Fig. 19 average value and median value of getattr operation run-time(GPFS)

ス数が増加するに従って getattr の処理時間が増加すると伺える。これは総プロセス数が増加するに従ってメタデータサーバへの競合が生じオーバーヘッドが増加するためと考えられる。同様の現象は GPFS でも見られ (図 16-17)、プロセス数を増加すると getattr 命令の処理時間が増加した。図 19 からもプロセス数と処理時間の相関関係が伺える。GPFS においてもメタノードへの競合が生じたことが原因と推定できる。また図 15 と図 17 を比較すると Lustre の方がメタデータアクセスが約 10 倍大きいことがわかる。これは複数ノードがメタノードとなるのでメタデータの負荷分散が行われるのに対し、Lustre では 1 台の MDT に対しメタデータアクセスが起こるので競合の影響がより大きいためと考えられる。

5. 関 連 研 究

科学データの増加と計算機の規模が巨大化するのに伴ってストレージへの I/O 性能を最大限に引き出すために様々な研究が行われている。近年は科学データを細粒度にわける必要性があることから、データサイズの小さいファイルが急増している。Carms らの研究[8] は並列ファイルシステムにおけるファイルサイズの小さいアプリケーションの I/O を削減するための手法を提案している。また分散環境におけるワークフローアプリケーションの実行の精査な解析というのはいくつも存在する。[9][10] 柴田らの研究[9] では五つのワークフローアプリケーションの I/O や計算時間を実行ログから解析している。またメタデータアクセスの最適化や大量に生成される中間ファイルをファイルシステムを介さずにタスク間で共有することの必要性が言及されている。分散環境におけるファイルシステムの性能解析として Juve らの研究[11] では Amazon EC2 上で様々な種類のワークフローを複数の共有ファイルシステム上で実行しクラウドにおけるワークフローアプリケーションの問題点を指摘している。

6. まとめと今後の課題

本稿では TSUBAME2.0 で稼働中の二つの並列ファイルシステムにおける細粒度な I/O 性能とメタデータアクセス性能を計測しその結果に対して考察を行った。まず IOzone を並列分散実行することで両ファイルシステムの基本的な I/O 性能を計測した。そしてデータインテンシブワークフローである Montage を実行し、実ワークフローアプリケーションの I/O の性能とメタデータアクセス性能を計測した。実験結果から我々の先の研究で現れた I/O 性能に影響を及ぼす 2 つの要素が確認された。計算ノード上のメモリのバッファキャッシュは両方のファイルシステムにおいて観測できたが、明確にスループットの差が現れた Lustre に対し GPFS ではスループットに大きく差が開くことはなかった。また並列ファイルシステムのメタデータアクセスについてはプロセス数が増加するとメタデータの競合が観測でき、単一のメタデータサーバにアクセスが集中した Lustre とメタノードで負荷分散がなされた GPFS で対照的なメタデータの挙動が確認できた。

今後の課題として、Lustre のストライピングのチャンクサイズなどファイルシステムの設定の変更や別のワークフローアプリケーションを用いてより詳細な性能解析を行う必要がある。また本稿から得られた知見をもとにワークフローアプリケーションの性能モデルを改良を行う。

謝辞 本研究の一部は JST CREST「ULP-HPC:次世代テクノロジのモデル化・最適化による超低消費電力ハイパフォーマンスコンピューティング」の援助による。

参 考 文 献

1) Dean, J. and Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters, *Operating Systems Design and Implementation (OSDI '04)*, pp.137–150 (online), available from ⟨http://www.usenix.org/events/osdi04/tech/dean.html⟩ (2004).
2) GXP : Grid and Cluster Shell: http://www.logos.ic.i.u-tokyo.ac.jp/gxp/.
3) 斎藤貴文, 千葉立寛, 佐藤仁, 松岡聡：ワークフローアプリケーションに対する計算資源割り当ての最適化, 第 129 回 HPC 研究会 (2011).
4) TSUBAME2.0: http://tsubame.gsic.titech.ac.jp.
5) Lustre: http://www.lustre.org/.
6) General Parallel File System: http://www-03.ibm.com/systems/software/gpfs/.
7) IOzone Filesystem Benchmark: http://www.iozone.org/.

Vol.2011-HPC-130 No.40
2011/7/28

8) Carns, P., Lang, S., Ross, R., Vilayannur, M., Kunkel, J. and Ludwig, T.: Small-file access in parallel file systems, *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, Washington, DC, USA, IEEE Computer Society, pp.1–11 (online), DOI:10.1109/IPDPS.2009.5161029 (2009).
9) Shibata, T., Choi, S. and Taura, K.: File-access patterns of data-intensive workflow applications and their implications to distributed filesystems, *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, New York, NY, USA, ACM, pp.746–755 (online), DOI:http://doi.acm.org/10.1145/1851476.1851585 (2010).
10) Dun, N., Taura, K. and Yonezawa, A.: Fine-Grained Profiling for Data-Intensive Workflows, *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, Washington, DC, USA, IEEE Computer Society, pp.571–572 (online), DOI:http://dx.doi.org/10.1109/CCGRID.2010.29 (2010).
11) Juve, G., Deelman, E., Vahi, L., Metha, G., Berriman, B., P.Berman, B. and Meachiling, P.: Data Sharing Options for Scientific Workflows on Amazon EC2, *In Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing (SC '10)*, pp.1–9 (2010).