Vol.2016-HPC-156 No.12
2016/9/16

「京」のファイルシステムの運用改善への取り組み

辻田　祐一[1,a)]　義崎 竜彦[2]　山本 啓二[1]　末安 史親[3]　宮崎 亮二[4]　宇野 篤也[1]

概要：スーパーコンピュータ「京」において，計算時のファイル I/O 性能を確保するために，計算開始前後にはプログラムやデータファイル等のファイルシステム間のステージング処理を行っている．これまでの運用の中で，大規模なデータファイル群のステージング処理中に，フロントエンドノード群から同じファイルシステムにアクセスする際のレスポンスが著しく低下する問題が散見されており，レスポンス向上に向けた運用改善を進めている．本稿では，このレスポンス向上を目指した運用改善に向けた取り組みとファイルシステムのパラメタ設定に関する性能評価について報告する．

キーワード：スーパーコンピュータ「京」，ステージング，FEFS，I/O ノード，ストライピング，負荷バランス制御

1. はじめに

スーパーコンピュータ「京」 [1]（以下，「京」）では，計算ノードでのファイル I/O の性能を確保するために，プログラムが実行される際に使用するファイルシステム（Local File System：以下，LFS）と，ユーザのプログラムやデータを格納するファイルシステム（Global File System：以下，GFS）で構成される 2 階層のファイルシステムを採用している．LFS と GFS の間は I/O 用のノード（以下，I/O ノード）を介して InfiniBand および「京」のノード間インターコネクトである Tofu [2] により接続されており，ステージング機構 [3, 4] により，GFS と LFS の間でファイルの転送処理を行っている．ステージング処理は他のジョブが実行されている最中に並行して行われるため，全体のジョブ実行効率の向上に大きく寄与している．

GFS に対するステージング処理に関しては，ジョブ開始前には GFS から LFS へのファイル転送を行うステージイン処理が，ジョブ終了後には LFS から GFS へのファイル転送を行うステージアウト処理が行われる．GFS は，このようなステージングでのアクセス以外にも，フロントエンドノード群からもアクセスされるが，大規模なデータのステージングが行われている最中に，フロントエンドノード群から GFS へアクセスする際のレスポンス低下が散見されている．これは，ステージング時の GFS へのファイル I/O による高負荷状態が原因であることが分かっている．また，ステージアウト処理では，保存先の GFS でのデフォルトのストライプカウント値が当初は 1 になっており，特定の OST（Object Storage Target）へのアクセス集中を招いていたことも大幅な性能低下の要因になっていた．フロントエンドノード群からのアクセスにおけるレスポンス低下は，ユーザの作業に大きな影響を与えるため，早急な改善が求められていた．

この問題に対し，我々はフロントエンドノード群から GFS へのアクセスにおけるレスポンス向上のために，ステージング処理とフロントエンドノード群からのアクセス処理の間の負荷バランス機能とファイルサイズに応じたストライプカウント設定を適用することで，レスポンス低下を低減させる試みを現在行っている．なお，現段階では過去の運用経験に基づいたパラメタ設定で運用を行っており，今後の運用改善に向けて，パラメタ設定を含む最適化が急務となっている．

そこで我々は過去の運用実績において大規模なステージング処理を行うアプリケーションを模擬して，最適なストライプカウント設定と負荷バランス設定の組み合わせを検証する評価試験を実施した．本稿では，この評価試験の実施に至った GFS への大規模データのステージング処理に伴うレスポンス低下の問題と，運用改善に向けた取り組みについて評価試験の分析も踏まえながら報告する．今回の評価試験からは，ストライプカウント設定と負荷バランス設定の有効性が確認できたと共に，この結果を踏まえた今後の運用改善の方向性が確認できた．本稿では，以下，第 2 章において，「京」のファイルシステムを中心としたシス

[1] 国立研究開発法人理化学研究所　計算科学研究機構
[2] 株式会社ナニワ計算センター
[3] 富士通株式会社
[4] 株式会社富士通システムズ・ウエスト
[a)] yuichi.tsujita@riken.jp

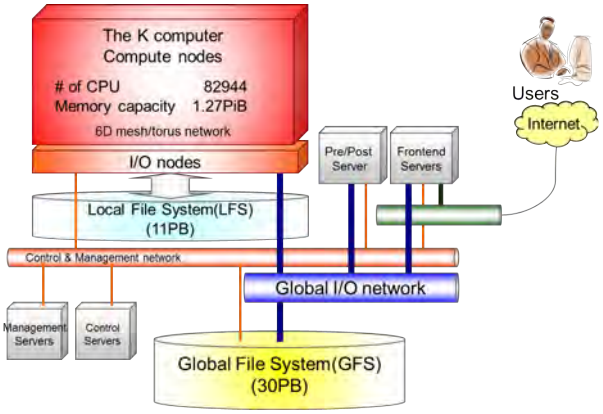


図 1 「京」のファイルシステムを含むシステム構成

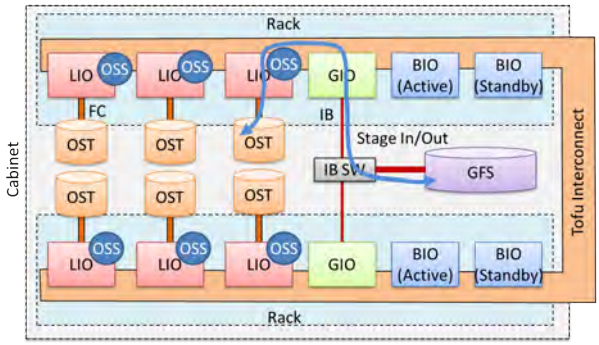


図 2 I/O ノード群と LFS および GFS の間のステージング処理のデータの移動の様子

テム構成とレスポンス低下の問題について述べ，第 3 章において，大規模データによるステージング処理による GFS のレスポンス低下を軽減する取り組みについて説明する．次に第 4 章において，運用改善に向けた性能評価結果について報告する．関連研究について第 5 章で述べた後に，第 6 章で本稿のまとめを行う．

2. 「京」におけるファイルシステムと GFS のレスポンス低下

本章では，「京」のファイルシステム構成について概略を説明した後に，これまでに散見された GFS のレスポンス低下の問題について説明する．まず始めに**図 1** に「京」のファイルシステムを含むシステム構成を示す．「京」は 1 ラックあたり 96 台の計算ノードを格納しており，全体で 82,944 台の計算ノードを有している．図 1 の”I/O nodes” と示したところには，I/O ノードと呼ばれる入出力やシステムファイルへのアクセスを行うためのノード群がある．この図に示すように 2 階層のファイルシステムを採用しているが，これを構成する LFS と GFS へのアクセスに用いる I/O ノードとしてそれぞれ LIO および GIO がある．LIO 及び GIO は各ラックにそれぞれ 3 台並びに 1 台ずつ配置されている．LFS 並びに GFS には，富士通が Lustre をベースに独自の機能拡張を行った FEFS（Fujitsu Exabyte File System） [5] が利用されている．運用上の理由から，GFS は複数のボリューム群で構成されているのに対し，LFS は全計算ノードからの一様なアクセスを実現するために 1 つのボリューム構成になっており，MPI プログラムでランク間で共有してアクセスする領域（共有ディレクトリ）を提供している．さらに LFS は，ランク毎の個々のファイル I/O の高速化のために，ランク毎に独立にアクセスできる領域（ランク番号ディレクトリ）が，loopback デバイスにより用意されている．いずれの LFS のアクセス領域も OSS（Object Storage Server）として使用されている LIO を通してアクセス可能になっている．なお，計算ノード群と LIO は Tofu で接続されており，RAID-50 で構築された LFS の OST 群を有するエンクロージャーと LIO 間は FibreChannel により接続されている．一方，LFS の MDS（Meta Data Server）が動作するノードは LIO を介して制御系のイーサネット（図 1 の”Control & Management network”）経由でアクセスできる．

図 2 に I/O ノード群と LFS および GFS の間で行われるステージング処理の例を示す．GFS に格納されているユーザのプログラムやデータファイルは，ステージイン処理により各 LIO 配下にある LFS の OST 内にある共有ディレクトリあるいはランク番号ディレクトリへコピーされた後にジョブが実行される．ジョブ終了後には LFS の OST 上に出力されたデータファイル等が GFS へステージアウト処理によりコピーされる．ここでファイルのコピー処理を行うのが GIO であり，計算ノード群や LIO とは Tofu で繋がっている．また GFS の MDS や OSS が動いているノード群は，この図の”GFS” 内に配置されているが，これらと GIO は InfiniBand（4×QDR）で繋がっているため，各 GIO は GFS および LFS の両者にアクセス可能になっている．GFS を構成する OSS ノード群と OST 群を形成する RAID-6 のディスクシステム群を格納するエンクロージャー間は FibreChannel で接続されている．

いくつかあるステージング処理の中で，ここでは我々が推奨しているランク番号ディレクトリを用いた方法でのステージング処理について説明する．ランク番号ディレクトリに関しては，それを使用するプロセスが配置されているキャビネット（Tofu の z 軸リンクを共有する 2 ラックで 1 キャビネットを構成）にある GIO のみから参照可能になっており，GFS とランク番号ディレクトリ間のステージング処理では，個々の対応する GIO がコピー処理を担う．この時にストライプカウントが 1 ならば，GFS へのアクセスは特定の OST に対するアクセスに，ストライプカウントが 2 以上に設定されていれば，指定した個数の OST 間でのストライピングアクセスが行われる．なお，ステージング処理は，GIO 単位では 1 ジョブに対して行われ，多重実行はされない．

一方で，GFS はフロントエンドノード群からもアクセス

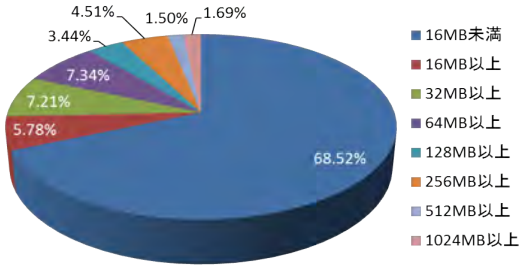


図 3 2015 年 10 月から 2016 年 3 月までの半年間の間にステージアウトまで実行された約 15 万個のジョブにおける各ジョブ毎の 1 ファイルあたりの平均サイズの内訳

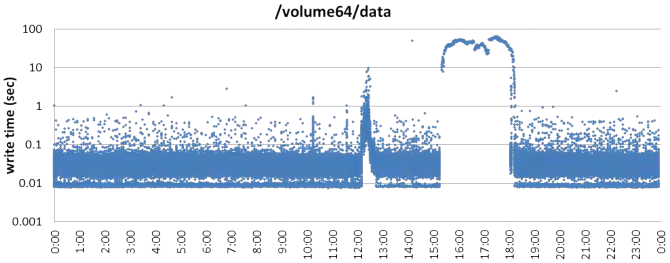


図 4 2015 年 2 月 5 日に発生した GFS へのアクセスのレスポンス低下の事例

されるが，この時に大規模なデータのステージング処理が行われている場合に，GFS へのアクセスのレスポンス低下が発生していた．ステージアウト処理でのファイルサイズの内訳の一例として，**図 3** に 2015 年 10 月から 2016 年 3 月までの半年間に「京」においてステージアウトまで実行された約 15 万個のジョブにおける各ジョブ毎の 1 ファイルあたりの平均サイズの内訳を示す．ステージアウトされるファイルの平均サイズに関しては，16 MB 未満が約 68%を占めており，16 MB 以上のファイルは全体の約 32%とそれほど多くは無く，128 MB 以上になると，全体の約 11%程度とさらに少なくなる．しかしながら，大きなファイルサイズを扱うジョブは一般に大規模なノード数を利用していることが多く，ステージング処理には時間を要するため，このような大規模データのステージングのタイミングに遭遇し，GFS へのアクセスでのレスポンス低下が発生する可能性は決して少なくはない．

次に，大規模データのステージングによりフロントエンドノードから GFS へのアクセス中にレスポンス低下が発生した事例を**図 4** に示す．この図では，2015 年 2 月 5 日の実際の運用情報から得られた GFS への 1 MB データの書込み時間の変化を示しているが，15 時過ぎから 18 時過ぎの約 3 時間の間に急激にレスポンス低下（アクセス時間の増加）が発生しているのが分かる．このタイミングにおいて，大規模（約 58 TB）かつ大量（約 24,000 ファイル）のファイル群がストライプカウントが当時のデフォルトの 1 でステージアウトされていたことが分かっている．ファイルサイズが小さい場合には，この設定で問題なかったが，この図の事例のように大規模データをストライプカウントが 1 でステージング処理を行っていた場合には，OSS や OST への負荷の偏りと高負荷を招き，偶発的に同じタイミングでフロントエンドノードから GFS にアクセスするユーザに悪影響を及ぼしていたケースが散見されていた．

このような GFS へのアクセスにおけるレスポンス低下の問題は，大規模なノード数を用いて大量のデータを扱うアプリケーションの利用頻度が高くなるにつれて，益々顕著になってゆくことが想定されるため，レスポンス向上に向けた運用改善が強く求められていた．

3. GFS レスポンス低下に対する運用改善

本章では，ステージング処理中におけるフロントエンドノード群から GFS にアクセスする際のレスポンス低下に対する運用改善の取り組み方法について，ステージアウト処理時の自動ストライプカウント設定と FEFS が有する QoS 機能によるファイルアクセス時の負荷バランス制御に関する現状と運用改善の方向性について述べる．

3.1 ストライプカウント設定

「京」で使用されている GFS はユーザのホーム領域及びスクラッチ領域を除き，各ボリュームは 12 台の OSS で構成され，各 OSS が 32 個の OST を有しており，一つのボリュームで 384 個の OST を有している．よってストライプカウントは最大で 384 になる．2 章で述べたように，過去にはユーザが指定しない限りストライプカウントが 1 でステージアウト先の GFS に書き込まれていた．

現段階では，ユーザ側で設定が行われていない限り，ファイルサイズを 16 MB で割った値と 32 の小さい方をストライプカウントとする方針で運用を進めている．この設定変更により，GFS アクセスの際のレスポンス向上に繋がっていることが確認されているが，このパラメタ設定は，過去の運用実績に基づいたものであり，さらなる運用向上の可能性がある．現状の設定では，使用する GIO の台数に関係無くファイルサイズのみに基づいてストライプカウントを設定しているが，使用する GIO の台数が多い場合には，GFS の OSS や OST が高負荷になり，レスポンス低下を招く可能性がある．よって更なる運用改善には，使用される GIO の台数，GFS の OSS の台数やステージング対象のファイル数などを考慮した適切なストライプカウント設定が必要と考えられる．

FEFS のベースとなる Lustre に関する入出力効率向上に向けたストライプカウントの設定については，現状では明確な対処法は無く，各々の運用サイトでのシステム構成や利用形態に応じて，経験的な側面からパラメタ設定をされているケースが多く見られる [6–8]．我々の場合も，運用経験から得られた知見からパラメタ設定を行うことが多

く，本稿での運用改善も同様な対応がなされてきた経緯があるが，多様なステージング処理に対して，できる限り最適な設定が行える枠組みの実現を目指している．

3.2 FEFS の負荷バランス制御

FEFS の I/O 処理において，OSS や MDS 上に起動されたサービススレッド群により処理が行われるが，クライアントから送られてくる I/O 要求に対し，通常は FCFS ポリシーに基づいた処理を行うため，複数のジョブの I/O 処理の中で，特に I/O 処理の負荷が高いジョブが I/O 処理を占有してしまう問題がある．また複数のユーザ間で I/O 処理の負荷が高いユーザに I/O 処理を占有される問題もある．FEFS では，これらの問題に対して QoS 機能を提供しており，I/O 要求元によって割り当てるサービススレッド数を制御する負荷バランス機能や，1 ユーザが発行できる I/O 要求の上限管理を行う機能が利用でき，大量に I/O 要求を発行するジョブによるファイル I/O 処理の占有を抑制することができる [5]．

我々の運用改善では，前者の機能を利用し，GFS の各 OSS のサービススレッド群に対し，フロントエンドノード群からのアクセスとステージング処理のそれぞれに割り当てられるスレッド数を制御する方法を用いている．現時点では，サービススレッド群の割り当てに関して，過去の運用実績から前者を最大スレッド数の 20%，後者を 80%として運用を進めているが，これもストライプカウントと同様に，更なる運用効率向上の可能性を検証する必要がある．

4. 性能評価

第 3 章で述べた現状と運用改善指針を踏まえ，「京」のファイルシステムの運用改善に向けて，最適なパラメタを策定する．そのために，我々は計算アプリケーションによる大規模データのステージング処理を模擬したプログラムを実際に「京」で走らせ，ステージング処理に対する GFS アクセスのレスポンスの評価を行った．使用した GFS は，運用管理上，一時的に共用利用から切り離されたもので，共用利用されている他の GFS と同じ機器構成であり，12 台の OSS を用いて 384 個の OST により構成されている．

この GFS に対し，1 個のジョブのステージング処理中に，1 台のフロントエンドノードから 5 分毎に dd コマンドにより 1 MB のデータの書込みを全 OST に対して行い，ステージング処理時のレスポンス低下を検証した．評価に用いたステージングを行うジョブのノード構成やステージング処理されるファイルサイズ等については，**表 1** に示した 3 パターンを用いた．上の 2 つでは，ストライプカウント設定と負荷バランス設定のそれぞれに関して評価を行うと共に，同じノード数（=プロセス数）で使用する GIO 数の違いによる振舞いの調査を行った．3 つ目は全 GIO（864 台）を使う条件下において，負荷バランス設定を行った上で，前者の評価から GFS レスポンス低下を最も小さくするストライプカウントと現行のストライプカウント値（32）で比較検討を行った．いずれの評価もストライプサイズはデフォルトの 1 MB を選択した．

表 1 試験に用いたノード数およびファイルサイズ

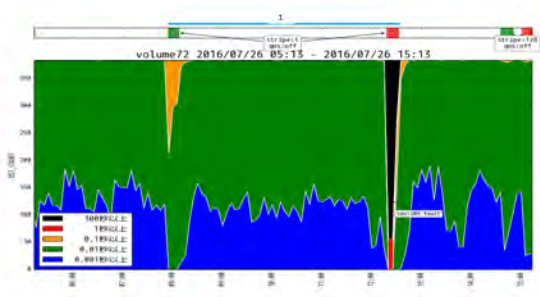
# nodes	Node layout	# GIO	File size
576	$12 \times 24 \times 2$	96	12 GB/file
576	$12 \times 12 \times 4$	48	12 GB/file
41,472	$48 \times 54 \times 16$	864	512 MB/file

本試験では，MPI プロセス群の各ランクで個々にファイル I/O を行うケースを想定し，ステージイン先としてはランク番号ディレクトリを用いた．よって各 GIO は同一キャビネット内にある担当するランク番号ディレクトリ内のファイルのステージング処理を行った．なお，ここで使用したデータファイルの大きさは，実際に「京」において大規模データのステージングを行うアプリケーションの一つである気象シミュレーションプログラム NICAM（Nonhydrostatic ICosahedral Atmospheric Model）[9] での利用実績を基に最初の 2 つのパターンでは 12 GB/ファイルの設定とした．一方，全 GIO を使う 3 番目のパターンでは，利用時間内に必要なデータを取ることを優先し，512 MB/ファイルの設定とした．全てのケースにおいて，プロセスあたり 1 個のファイルを割り当てて実施した．

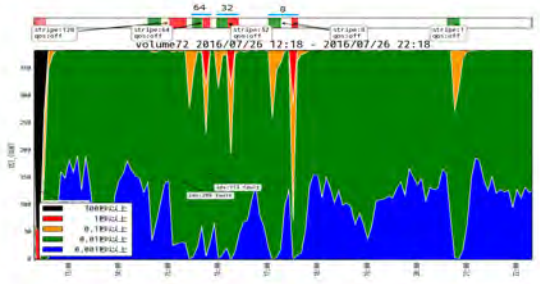
4.1 ストライプカウントの評価

表 1 に掲載した最初の 2 つのノード数に関して，負荷バランス制御無しの下で，ストライプカウントの設定を変えてフロントエンドノードからの dd による 1 MB データの書き込み時間の振舞いを確認した．ここで使用したストライプカウントは 1，8，32，64，128，256，384 の 7 パターンである．各々のパラメタで 2 回ずつ計測を行った．

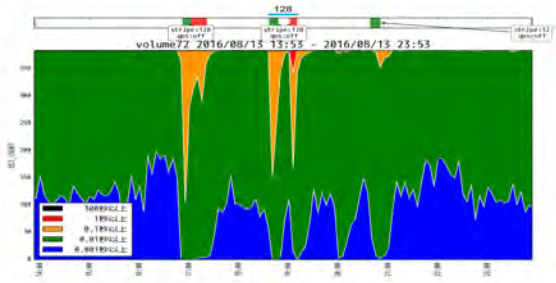
図 5 に $12 \times 24 \times 2$ ノード構成により 576 プロセスを起動して 96 台の GIO によるステージング処理を行った際の OST 毎のレスポンス時間の分布を示す．各々の図の上部にステージインとステージアウトが行われた時間帯をそれぞれ緑と赤の帯で示しており，評価対象のステージイン・ステージアウト間を青色の線でストライプカウント数と共に明示している．また図の縦軸は OST の累積数になっており，各々の時間でのレスポンス時間の分布を色分けして示している．青色が最もレスポンスの良い OST で，次第に緑，オレンジ，赤，黒の順にレスポンスが悪くなっていることを示している．ステージインに比べ，ステージアウトでのレスポンス低下が大きくなる傾向があるが，この結果から，ストライプカウントが 128 の時が最もレスポンス低下を低減できていることが分かる．なお，384 個の OST 間でストライピングされたファイルの累積サイズの分布について，ストライプカウントが 1，32 並びに 128 の時の様子



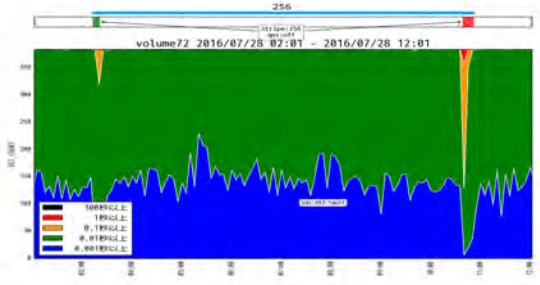
(a) ストライプカウント=1



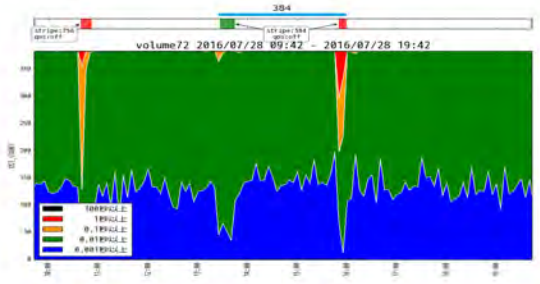
(b) 左からストライプカウント=64，32，8



(c) ストライプカウント=128

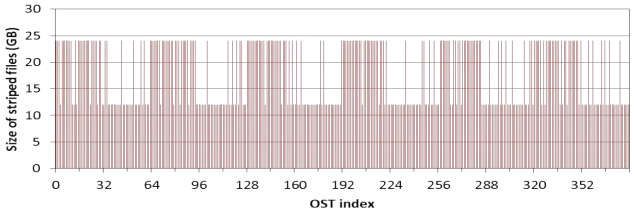


(d) ストライプカウント=256

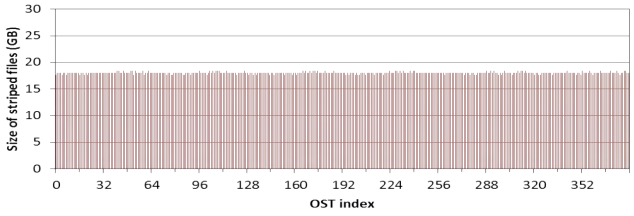


(e) ストライプカウント=384

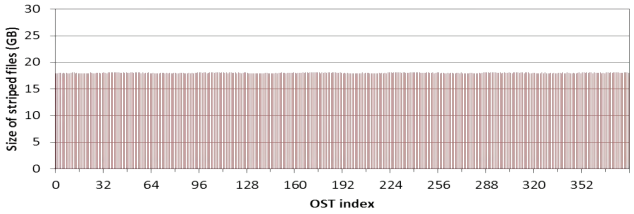
図 5 負荷バランス設定なしでのストライプカウントを変えた時のステージング処理時の GFS レスポンス時間の分布（96 台の GIO を使用）



(a) ストライプカウント=1



(b) ストライプカウント=32



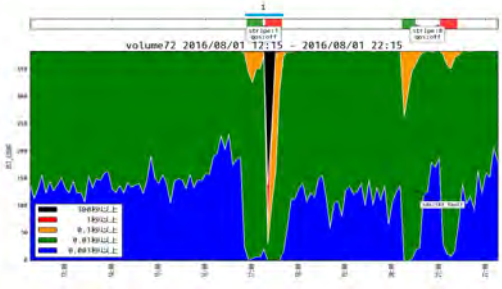
(c) ストライプカウント=128

図 6 384 個の OST 間でのストライピングされたファイルの累積サイズの分布

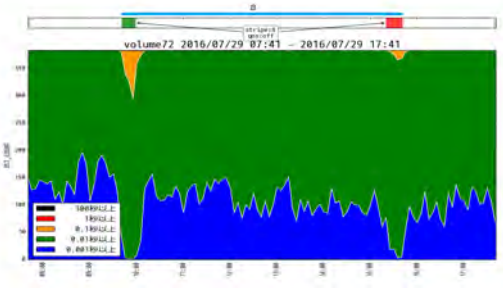
を**図 6** に示す．横軸は OST のインデックスで，縦軸はストライピングされたファイルの累計サイズになっている．ファイル数とストライプカウントの積が OST の数で割り切れない場合に負荷バランスが悪くなるが，この図からもストライプカウントが 1 の場合，OST 毎の受け持つ累積ファイルサイズに大きな偏りが生じ，I/O 処理の負荷バランスが悪かったことが分かる．一方，ストライプカウントを大きくするにつれて OST 毎の累積ファイルサイズが平滑化されるため，I/O 処理の負荷バランスが改善してゆくのが，ストライプカウントが 32 や 128 の時の様子から分かる．

次に GIO の台数を 48 台にした場合の GFS レスポンス時間の分布を**図 7** に示す．この結果と図 5 の 96 台の GIO を使用した時の結果を比較すると，後者の方がレスポンス低下の影響が大きいことが分かる．これは，I/O 要求を発行する GIO の台数が多いほど，GFS に対してより高負荷な状況を発生させていたためと考えられる．

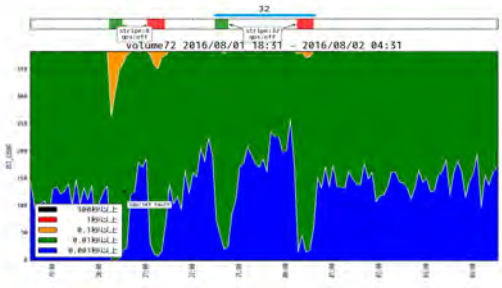
一方，各ストライプカウントでのステージング処理時間について 96 および 48 台の GIO での結果を**表 2** 並びに**表 3** にまとめた． まず，両者共に 1 回目と 2 回目のステージイン並びにステージアウトの時間にばらつきがあるが，これは割り当てられた OST の組合せによっては，通信やディスク I/O の混雑状況に加え，OST の RAID ディスク内の再構築処理が時々発生しており，これらによって多少変動



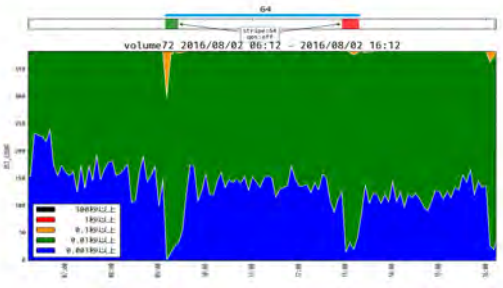
(a) ストライプカウント=1



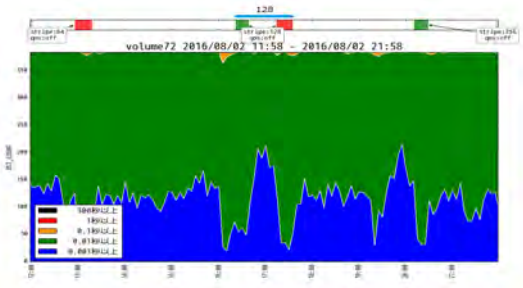
(b) ストライプカウント=8



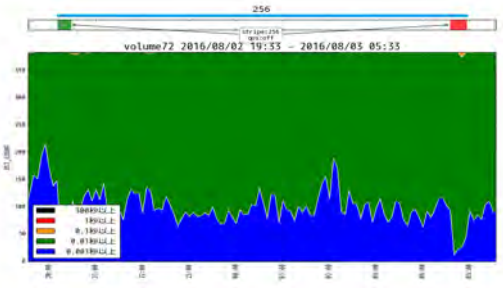
(c) ストライプカウント=32



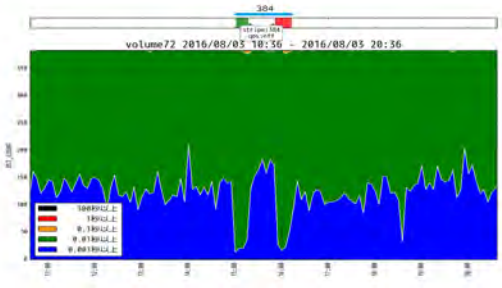
(d) ストライプカウント=64



(e) ストライプカウント=128



(f) ストライプカウント=256



(g) ストライプカウント=384

図 7 負荷バランス設定なしでのストライプカウントを変えた時のステージング処理時の GFS レスポンス時間の分布（48 台の GIO を使用）

表 2 96 台の GIO を用いたステージング処理時間（各ストライプカウントで，上段が 1 回目，下段が 2 回目）

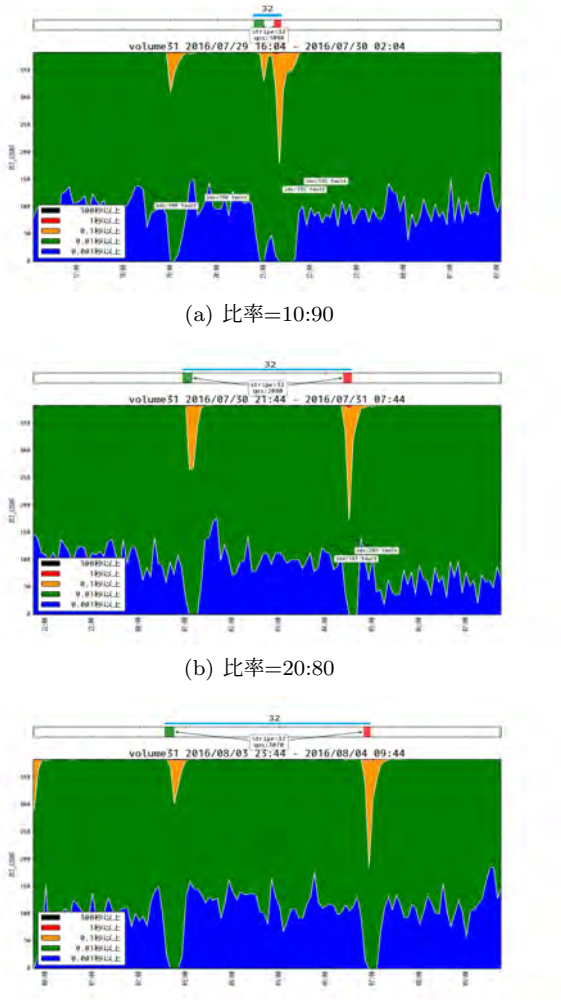
Stripe count	# GIO	SIN（秒）	SOT（秒）
1	96	776	838
	96	850	851
8	96	751	470
	96	856	426
32	96	711	416
	96	774	726
64	96	707	598
	96	686	472
128	96	571	1069*1
	96	686	483
256	96	521	752
	96	457	731
384	96	437	521
	96	996	506

表 3 48 台の GIO を用いたステージング処理時間（各ストライプカウントで，上段が 1 回目，下段が 2 回目）

Stripe count	# GIO	SIN（秒）	SOT（秒）
1	48	1040	1380
	48	1171	1221
8	48	1005	1205
	48	964	1292
32	48	941	1467
	48	983	1215
64	48	978	1216
	48	953	1273
128	48	984	1242
	48	937	1169
256	48	941	1229
	48	1001	1217
384	48	945	1284
	48	896	1198

した為と思われる．また測定のタイミングによっては，1 台の GIO で通信の不具合による処理時間の増加の影響があった．2 回の処理時間の平均値で比較したところ，全般的にストライプカウントが 1 では，ややステージアウト処理の時間が長くなるような傾向が見られるが，処理時間の変動等も考えられるため，今回の評価結果からは，ステージング処理時間に関して有意な差は確認できなかった．

なお，表 2 並びに表 3 に示す結果から見積もられる 1 台の GIO あたりのステージアウト性能を調べると，両者ともほぼ同じ（約 130 MB/s）である．使用している GFS の構成は同じため，GFS 側で律速していたのではなく，それ以外で律速していたと考えられる．使用した LFS のランク番号ディレクトリでのファイル I/O が同等の性能であることから，この環境では，LFS からのファイルコピー処理で律速されていたものと思われる．よって，両者ともステージング対象のファイルサイズは同じであるため，GIO の台数を 96 から 48 に減らしたことにより，ステージング処理の時間が長くなったものと考えられる．



(a) 比率=10:90

(b) 比率=20:80

(c) 比率=30:70

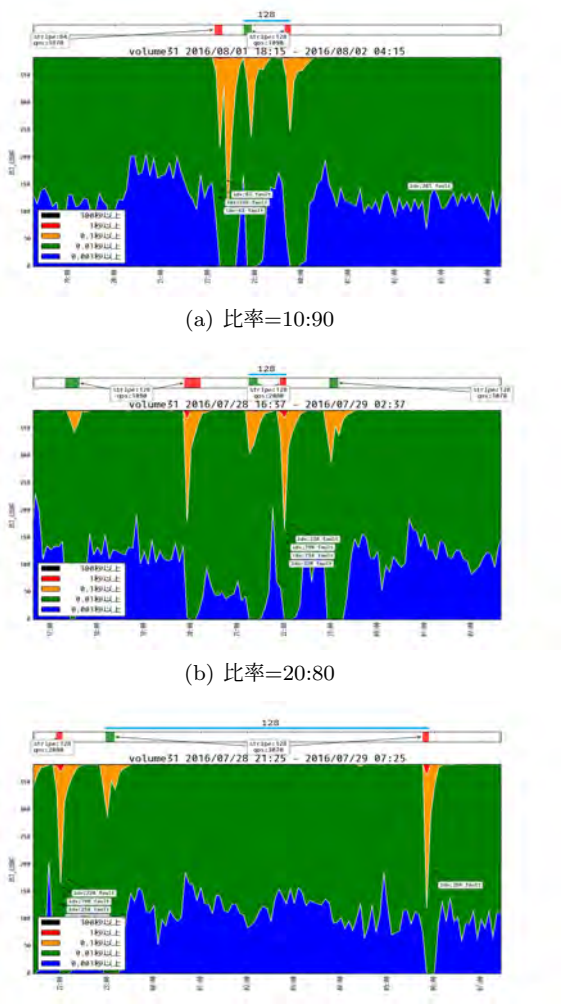
図 8 負荷バランス設定ありでのステージング処理時の GFS レスポンス時間の分布（96 台の GIO 使用，ストライプカウント=32）

4.2 負荷バランス設定の評価

次に，同じノード・ファイルシステム構成により，負荷バランス制御の効果を検証する評価を行った．まず最初に 96 台の GIO を用いたケースにおいては，現在の運用方式での最大ストライプカウントである 32 と，GFS レスポンスが最も良くなったストライプカウントが 128 の 2 パターンで，負荷バランス設定を有効にした場合の GFS レスポンス時間並びにステージング処理時間について評価を行った．

図 8 及び**図 9** にストライプカウントが 32 並びに 128 の時の計測結果を示す． 両方の図において，負荷バランス設定はフロントエンドノードからのアクセスとそれ以外からのアクセスの比で記述している．ストライプカウントが 32 の場合，図 8 に示した負荷バランス設定を適用した方が，適用しなかった図 5 での結果と比べて，レスポンス低下が

*1 1 台の GIO の InfiniBand 通信のシステム障害による遅延



(a) 比率=10:90

(b) 比率=20:80

(c) 比率=30:70

図 9 負荷バランス設定ありでのステージング処理時の GFS レスポンス時間の分布（96 台の GIO 使用，ストライプカウント=128）

低減されている．図 9 のストライプカウントが 128 の場合でも，負荷バランス設定が無い図 5 の結果と比べてレスポンス低下の低減が見られる．よって，負荷バランス設定は GFS へのレスポンス向上に有効な手法と考えられる．

次に，ステージング処理時間を**表 4** に示す．表 2 の負荷バランス設定無しで同じストライプカウントでの処理時間と比較しても，システム起因の問題が発生した場合を除いては，負荷バランス設定によるステージング時間の有意な増減は見られなかった．また 3 パターンの比率で有意な差は見られなかった．

以上の結果を踏まえ，「京」の全 GIO（864 台）を用いたステージング処理による GFS アクセスのレスポンス評価をこれまでと同様の手法で行った．$48 \times 54 \times 16$ の構成で 41,472 ノードを用い，1 ノードあたり 1 プロセスを起動してステージング処理を行った．既に説明したように，この試験では 1 つのファイルの大きさを 512 MB にしており，利用時間の関係から各々のケースで 1 回づつ評価した．また，FEFS の負荷バランス設定に関しては，前述の評価での 3 パターンでの比率での有意な差が確認できなかったため，これまでの運用実績を基に，フロントエンドノード側を 20%，それ以外を 80%に設定した．その時の dd コマンドによる 1 MB のデータの書込みに要した時間の分布を**図 10** に示す．ストライプカウントについては，現行のストライプカウント設定との比較や，使用する GFS の OSS の台数との兼ね合いから，この評価では，ストライプカウントとして，現行の運用ポリシーの下で今回のファイルサイズで設定される 32 に，6 並びに 12 を加えた 3 パターンで評価した．ストライプカウントが 32 や 6 の場合と比較して，ストライプカウントが 12 の時に GFS アクセスにおけるレスポンス低下が低減されている．

各々のストライプカウントでステージングに要した時間を**表 5** に示す．ステージインに関しては，ストライプカウントが大きくなるほど各 GIO で対応する OST の数が増え，高負荷状態を招き，ステージイン時間の増加に繋がった可能性も考えられるが，今回は利用時間の制約から各パラメタで 1 回のみの計測であるため，処理時間の変動の可能性も排除できず，判断は難しい．同様に，ステージアウトに関しても，ストライプカウントの増加に伴い，OSS や OST 間の負荷バランスが改善されて処理時間の短縮に繋がった可能性も考えられるが，やはり判断は難しい．適切なストライプカウント設定については，今後，さらに検証を進める予定である．

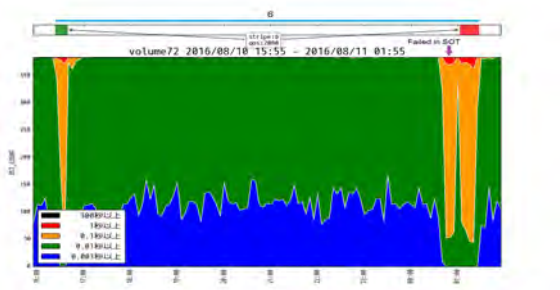
表 4 負荷バランスを有効にした際のステージング処理時間（96 台の GIO を使用）

Stripe count	比率	SIN（秒）	SOT（秒）
32	10:90	733	460
		653	471
	20:80	679	574
		708	458
	30:70	674	465
		711	441
128	10:90	545	619
		802	797
	20:80	574	492
		695	497
	30:70	1069	504
		595	449

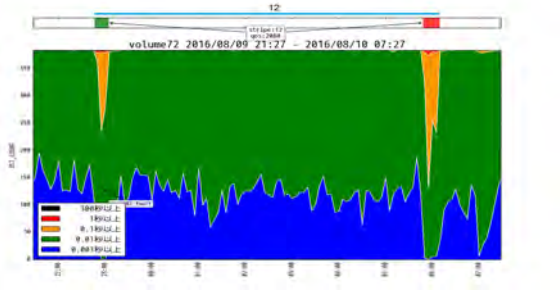
表 5 全 GIO を用いたステージング処理に要した時間

Stripe count	# GIO	SIN（秒）	SOT（秒）
6	864 (SIN 時は 863)*2	910	1433
12	864	998	1218
32	863*2	1130	1168

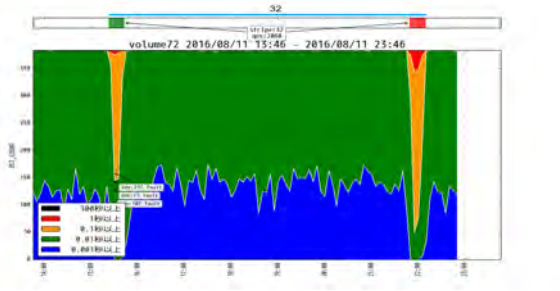
*2 システム障害発生に伴う GIO 台数減少



(a) ストライプカウント=6



(b) ストライプカウント=12



(c) ストライプカウント=32

図 10 全 GIO によるステージング処理における GFS レスポンス時間の分布（図 10(a) のストライプカウントが 6 の時にはシステム障害による SOT のリトライが行われている.）

5. 関連研究

Lustre における I/O 処理の QoS 制御方式については，Yingjian ら [10] が，クライアントからの I/O 命令に対し，アクセスするデータ領域などに基づいたシーク操作が少ない RPC リクエスト処理と，デッドラインスケジューリングの 2 つを Network Request Scheduler (NRS) として提案している．また，NRS に対していくつかの QoS ポリシーが提案されており，最近では Token Bucket Filter を用いた QoS ポリシーが提案されている [11]．これらの QoS 機能はサーバへの RPC 要求発行制御を行う実装になっている．一方，FEFS では RPC 要求の送信元に応じて，OSS や MDS での RPC 要求に対するサービススレッドの割り当て数を制限して I/O 処理の負荷バランスの調整を行う機能を有しており，NRS で実装されている QoS 機能とは実装方法が異なる．

ネットワーク越しに並列ファイルシステムにアクセスする大規模なデータのステージングのような処理は，同じネットワークを使う通信に少なからず影響を与えるために，様々な性能低下の低減手法が行われている [12, 13]．Rajachandrasekar ら [12] は，ステージング処理時に同じ InfiniBand 接続網を用いる MPI 通信への影響を InfiniBand の QoS 機能を用いて低減している．ここでは，MPI 通信側に高い優先度を与え，通信経路が空いている間のみステージング処理を行わせることで，通信経路の混雑を回避し，MPI 通信性能を確保している．他にも，Zhang ら [13] は，PVFS2 [14] に対して，機械学習を用いてユーザからのジョブ実行時間の要求に基づいた I/O 性能の上限値を求め，QoS によるファイル I/O を実現させている．一方，我々の取り組みでは，ファイルシステムへのアクセスにおける負荷バランス制御を実現させている．これは，大規模なデータアクセスにより，他のファイルアクセスのレスポンス低下を低減させることを目的としており，上で述べたような取組みとは目的や方法が異なる．

6. 本稿のまとめ

本稿では，大規模データのステージング中におけるフロントエンドノード群から GFS へのアクセスのレスポンス向上を目指した運用改善への取り組みにおいて，ストライプカウントや FEFS の負荷バランス機能に関して性能評価を行った．既に両者の設定に関して過去の運用経験から得られたパラメタを設定してレスポンス向上に努めているが，更なる運用改善に向けて，今回の性能評価からより適切な設定方法の検討を行った．

ストライプカウント設定と FEFS が有する負荷バランス設定に関して最適化の検証試験を行ったところ，OST 間で負荷バランスが良くなるように，適切なストライプカウントを設定することが，GFS レスポンス向上において重要であることが確認された．ただし，必要以上にストライプカウントを大きくすることはファイルシステムの高負荷に繋がる可能性があるため，ステージングで使用される GIO の台数や GFS の OSS の台数に加え，ステージング対象のファイル数やファイルサイズを考慮したストライプカウントの設定が有効になると期待される．次に FEFS の負荷バランス設定についても，GFS アクセス時のレスポンス向上に有効であることが確認できた．以上により，特に大規模なノード数を用いたジョブのステージング処理に対しては，適切なストライプカウントと負荷バランス設定を行うことで，レスポンス低下を可能な限り小さく抑えつつ，一方でステージング処理時間の増加を出来る限り招かない運用が期待される．またステージング処理における GIO の台数が少ないほど，GFS でのレスポンス低下が低減できるため，大規模なステージングが予想されるジョブに対しては，GIO の台数を少なくするノード構成を適用させる運用の可能性も考えられる．

ただし，今回の評価では，負荷バランス設定が行われている場合におけるストライプカウント設定の有効性に関する検証を行っていない．負荷バランス設定機能により，適切にステージング処理とフロントエンドノードからのアクセスのスレッド割り当てを行っても，ストライプカウントが 1 の設定では，特定の OST への高負荷状態を回避できないため，適切なストライプカウント設定と共に用いることが必須と考えている．これについては，今後検証を進める予定である．なお，FEFS が有する負荷バランス設定には，休んでいるサービススレッドがある場合の負荷バランスの最大許容値を追加で設定できる機能がある．これを利用することで，ステージング処理あるいはフロントエンドノードからのアクセスのいずれかが少ない，あるいは無い場合に，もう一方の処理に割り当てられるスレッド数を動的に増やせるため，サービススレッド群の有効利用が可能になる．今後，パラメタ設定に関わる指針の策定に向けた検証等を進めると共に，この機能も含めたより効果的な運用改善を進める予定である．

参考文献

[1] 特集：スーパーコンピュータ「京」，情報処理， Vol. 53, No. 8, pp. 752–807 (2012).
[2] Ajima, Y., Inoue, T., Hiramoto, S., Takagi, Y. and Shimizu, T.: The Tofu Interconnect, *IEEE Micro*, Vol. 32, No. 1, pp. 21–31 (2012).
[3] 宇野篤也，庄司文由，横川三津夫：ファイルステージングのあるジョブスケジューリングの評価，情報処理学会研究報告，Vol. 2012-HPC-136, No. 22 (2012).
[4] 山本啓二, 宇野篤也, 塚本俊之, 菅田勝文, 庄司文由：スーパーコンピュータ「京」の運用状況，情報処理，Vol. 55, No. 8, pp. 786–793 (2014).
[5] Sakai, K., Sumimoto, S. and Kurokawa, M.: High-Performance and Highly Reliable File System for the K computer, *Fujitsu Sci. Tech. J.*, Vol. 48, No. 3, pp. 302–309 (2012).
[6] Saini, S., Rappleye, J., Chang, J., Barker, D., Mehrotra, P. and Biswas, R.: I/O Performance Characterization of Lustre and NASA Applications on Pleiades, *High Performance Computing (HiPC), 2012 19th International Conference on*, pp. 1–10 (2012).
[7] Ezell, M., Mohr, R., Wynkoop, J. and Braby, R.: Lustre at Petascale: Experiences in Troubleshooting and Upgrading, *2012 Cray User Group Meeting (CUG)* (2012).
[8] Crosby, L. D. and Mohr, R.: Petascale I/O: Challenges, Solutions, and Recommendations, *Proceedings of the Extreme Scaling Workshop*, BW-XSEDE '12, University of Illinois at Urbana-Champaign, pp. 7:1–7:7 (2012).
[9] Satoh, M., Matsuno, T., Tomita, H., Miura, H., Nasuno, T. and Iga, S.: Nonhydrostatic icosahedral atmospheric model (NICAM) for global cloud resolving simulations, *Journal of Computational Physics*, Vol. 227, No. 7, pp. 3486–3514 (2008).
[10] Yingjian, Q., Barton, E., Wang, T., Puntambekar, N. and Dilger, A.: A Novel Network Request Scheduler for a Large Scale Storage System, *Computer Science - Research and Development*, Vol. 23, No. 3, pp. 143–148 (2009).
[11] Ihara, S.: A New Quality of Service (QoS) Policy for Lustre Utilizing the Lustre Network Request Scheduler (NRS) Framework, *Lustre Administrator and Developers Workshop (LAD'13)* (2013).
[12] Rajachandrasekar, R., Jaswani, J., Subramoni, H. and Panda, D. K.: Minimizing Network Contention in InfiniBand Clusters with a QoS-Aware Data-Staging Framework, *2012 IEEE International Conference on Cluster Computing*, pp. 329–336 (2012).
[13] Zhang, X., Davis, K. and Jiang, S.: QoS Support for End Users of I/O-intensive Applications Using Shared Storage Systems, *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, ACM, pp. 18:1–18:12 (2011).
[14] Latham, R., Miller, N., Ross, R. and Carns, P.: A Next-Generation Parallel File System for Linux Clusters, *LinuxWorld*, Vol. 2, No. 1 (2004).