Vol.2014-HPC-145 No.34
2014/7/30

WebGfarm: 分散ファイルシステムGfarmのHTTPインタフェース

鷹津 冬将[1,3,a)]　建部 修見[2,3]

概要：膨大なデータを管理するために用いられる分散ファイルシステムにおいては．POSIX 準拠のインタフェースが提供されることが一般的である．しかしながらビッグデータ分野で用いられる分散ファイルシステムにおいては，データの読み書きを行うためにインターネットで広く使われている HTTP を使った REST インタフェースを提供する分散ファイルシステムも登場してきた．本稿では，広く使われている HTTP を用い分散ファイルシステム Gfarm 上のデータにアクセスするインタフェースを設計し、Apache HTTP Server 向けの Gfarm モジュールを実装、アクセス性能の初期評価を行った。評価においては，WebGfarm がプロセス数に比例して性能が向上することや，データローカリティを考慮することによりファイルサイズなどの条件次第でアクセス性能が向上することを確認した．

1. はじめに

データインテンシブコンピューティングや e-サイエンスの分野においては高性能計算機が用いられる．このような高性能計算機においてデータはサイズが非常に膨大であり，ひとつのストレージデバイスにおいてデータを管理することは出来ないため，Gfarm [1] などの分散ファイルシステムを用いることにより複数のストレージデバイスによって管理することが一般的である．この分散ファイルシステムで管理されるデータも，近年の実験装置の発展やシミュレーション規模の拡大などにより，増加の一途をたどっており，管理されるデータの合計サイズもペタバイト級となっている．合計データサイズも年々増加しているため，数年後にはポストペタ，すなわちエクサバイト級やゼタバイト級に達することが想定できる．そのため分散ファイルシステムもこうした超巨大なデータを処理する必要がある．

こうした課題に対して，筆者らはこのような大規模なデータを処理することのできる分散ファイルシステムを実現することを目指しており，これまでに分散ファイルシステムのストレージサーバで利用されることを想定した不揮発性デバイス向けのオブジェクトストレージを設計し発表してきた．分散ファイルシステムは一般的にファイルの中身を管理するストレージサーバと，ファイルのメタデータについて管理するメタデータサーバによって構成されるが，これらだけでなくファイルに対してアクセス出来るようにするランタイムやインタフェースも重要な要素である．分散ファイルシステムにおいては，FUSE [2] 等を利用し POSIX 準拠のインタフェースを備えるのが一般的であり，Gfarm の場合においては libgfarm を利用し FUSE を経由するファイルシステム gfarm2fs を提供している．libgfarm はプログラムから直接 Gfarm ファイルシステムを操作できるライブラリであり，このライブラリを利用することで他のインタフェースを実装することが出来る．libgfarm を利用した例として MPI-IO 向けのインタフェース [3] や Hadoop 向けのインタフェース [4] がある．しかしながら，FUSE を経由したファイルシステムでは，同一ディレクトリに多数のファイルを同時に作成する場合にディレクトリエントリがロックされるため性能が性能が伸びない．また，libgfarm を用いたアプリケーションを作成するには Gfarm に関する知識を要する．

近年のビッグデータ分野で用いられる分散ファイルシステムである HDFS [5] や Ceph [6] においてはデータの読み書きを行うために HTTP を使った REST [7] インタフェースが用意されている．WEB 等において広く使われている HTTP を用いることで，その分散ファイルシステムに関する知識がなくても分散ファイルシステムに対しアクセスを行うプログラムを作成することが可能となった．REST インタフェースを用いることで，サーバ台数を増やすことによる性能の向上も見込めるが，この場合においては，データローカリティによる性能への影響が課題となる．例えば

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 筑波大学システム情報系
3 独立行政法人科学技術振興機構 CREST
a) takatsu@hpcs.cs.tsukuba.ac.jp

© 2014 Information Processing Society of Japan

複数台のサーバを用意した場合にデータのあるホストへリダイレクトを行うことでネットワーク上を流れるデータ量が減り，性能が向上する可能性がある．一方で，リダイレクトに伴いクライアント側で再度データのあるホストに対してリクエストを発行し直さなければいけないと言ったオーバヘッドがある．Ceph が提供している REST インタフェースではこういったデータローカリティについては考慮を行っていない．

本稿では分散ファイルシステム Gfarm において，これまで実装されてこなかった HTTP インタフェースである WebGfarm を設計・実装を行い，初期評価を行った．WebGfarm においては，ローカルへのアクセスを行った際に高い性能を発揮するという Gfarm の特長を活かすためにデータローカリティを考慮したモードと，HTTP のリクエストのオーバヘッドによる影響を減らすために Gfarm プロクシとして動作するモードの二つのモードを設計した．初期評価では，アクセス性能の評価として同一ディレクトリにおいて多数のファイルを作成し，ファイルへの操作を行う評価や，データローカリティを考慮した場合における性能への影響について評価を行った．

この論文による貢献は以下の 2 点である．

- 分散ファイルシステム Gfarm の HTTP インタフェースの設計・実装
- データローカリティを考慮することによる影響の調査

本稿は 5 章で構成される．第 2 章では，関連研究について述べる．第 3 章では，本研究で設計した WebGfarm の詳細について示す．第 4 章では，WebGfarm の実装について詳述する．第 5 章では，評価実験の詳細及び評価結果を示す．そして第 6 章で，結論と今後の課題及び展望について示す．

2. 関連研究

HTTP を経由した REST API を用いることで処理を行うテクニックは広く用いられている．HTTP を通じ REST インタフェースを用意している分散ファイルシステムとしては，先にも述べたとおり Ceph と HDFS がある．

Ceph においては Amazon S3 や OpenStack Swift [8] と互換性のあるインタフェースを備えている．これらのインタフェースは Ceph のライブラリである librados を用いた Object Gateway である．この Ceph Object Gateway は CGI プログラムとして書かれており，HTTP のパーサーなどは Apache 等の Web サーバーを利用するプロクシ型のインタフェースである．サーバー台数を増やすことで性能もスケールするが，データローカリティについて考慮は行っていない．

一方で，HDFS の REST インタフェースは 2 種類あり，WebHDFS [9] と呼ばれるものと HttpFS [10] と呼ばれるものがある．WebHDFS は HDFS に組み込まれた REST HTTP API を提供するインタフェースである．WebHDFS においては URL 内にファイルのパスと，操作の名前を指定する．リクエストを送るホストはそのリクエストに適したホストへリダイレクトされる．たとえばファイルを開いて読み込むリクエストを行う場合は，ネームノードにリクエストを送ったとしても，そこからファイルのデータがあるデータノード上の WebHDFS へリダイレクトされる．HttpFS は WebHDFS と共通の REST HTTP API を提供する HDFS プロクシ型のインタフェースである．WebHDFS とは違い，HttpFS は Java で書かれたウェブアプリケーションである．これも Ceph Object Gateway と同様にサーバー台数を増やすことで性能をスケールさせることが可能であるが，データローカリティについて考慮は行っていない．

表 1 WebGfarm の API

目的	用途
ファイルの作成	PUT /Filename
ディレクトリの作成	PUT /Dirname/
ファイルへの書き込み	PUT /Filename
ファイルへの追記書き込み	POST /Filename
ディレクトリの読み込み	GET /Dirname/
ファイルの読み込み	GET /Filename
ファイルの削除	DELETE /Filename
ディレクトリの削除	DELETE /Dirname
メタデータの取得	HEAD /Entryname

3. WebGfarm の設計

WebGfarm は分散ファイルシステム Gfarm の Web フロントエンドとして設計した．HTTP はメソッドとして GET,PUT,POST,DELETE を利用することが出来る．これらを用いて API を設計した．ファイル・ディレクトリに関する API を表 1 に示す．

表 1 に示される各 API を HTTP でリクエストを行うことでファイルやディレクトリの操作を完了させる．各操作をより簡潔な表現で行うために，各操作をファイル名とメソッドによって指定している．

3.1 二つのモード

WebGfarm は性能を向上させるために，複数のサーバで運用することを想定している．Gfarm のストレージノードデーモンである gfsd が稼働するホストに HTTP でリクエストを行った際に，そのリクエストしたファイルがそのホスト上に存在するとは限らない．Gfarm ではローカルのストレージノードでファイルがある場合に高い性能があるという特徴があるため，WebGfarm ではファイルがあるストレージノードへリダイレクトをするダイレクトアクセスモードを用意する．一方で，すべてのストレージノードで

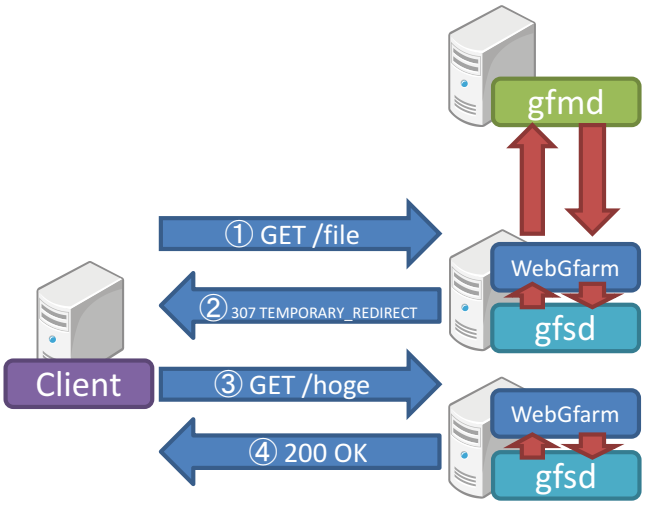

図 1 ダイレクトアクセスモードの動作モデル

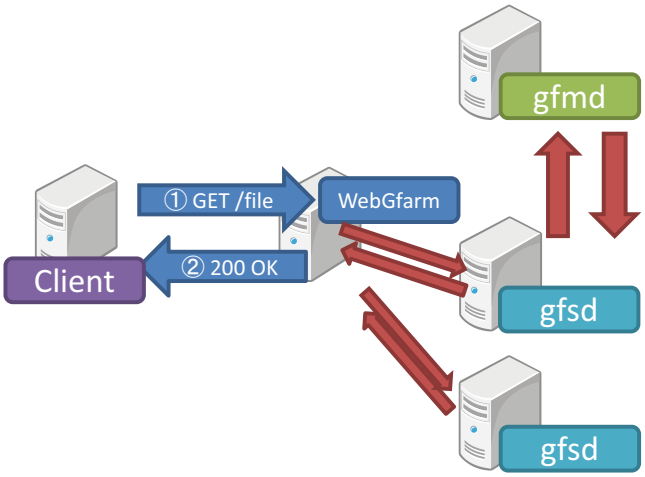

図 2 プロクシモードの動作モデル

WebGfarm を動作させることが難しい場合もある．そのため，HttpFS のように Gfarm Proxy として動作するプロクシモードも用意する．それぞれの動作モデルを図 1 及び図 2 に示す．

図 1 に示されるダイレクトアクセスモードでは，リクエストに応じて，ファイルがどのストレージノードにあるか検索する．ファイルがあるノードのうち適したノードを選択し，そのノードへリダイレクトを発行する．クライアントはこのリダイレクトをもとに，データのあるノードへリクエストを再発行する．

一方で，図 2 に示されるプロクシモードでは，リクエストに応じて直接 WebGfarm がストレージノードからデータをコピーし，それをクライアントに転送する．この場合では，ストレージノードからデータコピーが 1 回増えるが，クライアントが行うリクエストは 1 回となり，HTTP のリクエストを発行することに伴う性能低下の影響を減らすことが出来る．また，プロクシモードは HttpFS と同様のプロクシ型のインタフェースとなっていることから，必ずしも Gfarm 上のすべてのノードにおいて WebGfarm を立ち上げる必要がない．

表 2 実装に用いた libgfarm の API

カテゴリ	名前	用途
ファイル操作	gfs_pio_create	ファイルの作成
	gfs_unlink	ファイルの削除
	gfs_pio_open	ファイルを開く
	gfs_pio_stat	ファイルの stat 構造体の取得
	gfs_pio_close	ファイルを閉じる
	gfs_pio_read	ファイルから読み込む
	gfs_pio_write	ファイルに書き込む
ディレクトリ操作	gfs_mkdir	ディレクトリの作成
	gfs_rmdir	ディレクトリの削除
	gfs_opendirplus	ディレクトリエントリを開く
	gfs_readdirplus	ディレクトリエントリからの読み込み
	gfs_closedirplus	ディレクトリエントリを閉じる
スケジューリング	gfarm_schedule_hosts_domain_by_file	ファイルがある gfsd の探索
	gfarm_schedule_hosts_acyclic	リダイレクトに最適な gfsd の探索
その他	gfarm_initialize	libfarm の初期化
	gfarm_terminate	libgarm の終了処理

4. 実装

これらの設計に基づきファイルの作成やデータの読み書きを HTTP を用いて行うプロトタイプを実装した．このプロトタイプ実装は，簡単のため mod_webgfarm という Apache HTTP Server [11] のモジュールという形で実装を行った．Apache HTTP Server のモジュールとすることで，HTTP のパーサー部分などの機能を利用できる．

今回のプロトタイプ実装では，ファイルの作成・削除及び読み書き，ディレクトリの作成・削除及びディレクトリ内のファイルリストの取得が可能となっている．

4.1 libgfarm

プロトタイプを実装するにあたり，WebGfarm から Gfarm への通信を行うライブラリとして libgfarm を用いた．今回のプロトタイプ実装で用いた libgfarm の API を表 2 に示す．

今回のプロトタイプ実装では，ファイルの作成・削除及び読み書き，ディレクトリの作成・削除及びディレクトリ内のファイルリストの取得が可能となるように実装したため利用した libgfarm の API もそれに準ずるものとなっている．

2 において示されている API うちスケジューリングの関

数はダイレクトアクセスモードの際に用いる．ダイレクトアクセスモードの際においては，ファイルのデータが保存されているストレージノードへ読み書きの際にリダイレクトを行う必要があり，これを実現するために用いた．この関数はファイルを作成するときにも呼ばれるため，ファイル作成時においてオーバヘッドとなり得る可能性がある．

4.2 mod_webgfarm

先述したとおり, mod_webgfarm は Apache HTTP Server のモジュールである．そのため，HTTP のリクエストは Apache HTTP Server が受信する．そして mod_webgfarm が，Apache HTTP Server が受けたリクエストに対して libgfarm を通じて Gfarm 上のファイルやディレクトリの操作を行い，結果を Apache HTTP Server に返す．Apache HTTP Server はリクエストに応じて，mod_webgfarm から返ってきたデータを加工してリクエスト発行元に結果を返す．加工の例として，Range ヘッダに対する処理がある．HTTP では Range ヘッダを用いることで，データ中の一部分のみ取得するようなリクエストを発行することができる．このようなリクエストに対しても一部のデータのみ返す処理については Apache HTTP Server が行うため，mod_webgfarm は Apache HTTP Server にリクエストされたファイルの内容すべてを返す．

mod_wegfarm がダイレクトアクセスモードとプロクシモードのどちらを使うかといった設定は Apache HTTP Server の設定ファイル上で行う．ダイレクトアクセスモードの時，mod_wegfarm は libgfarm のスケジューリングに関する関数を使いデータのある gfsd のアドレスやそれらの CPU 使用率から適したホストへリダイレクトを行う様に 307 Temporary Redirect を返す．その際，適したホストが自ホストの場合においてはリクエストに応じて処理を行い，結果を返す．一方プロクシモードの場合においては，データが自ホスト上の gfsd にあるかどうかに関わらず libgfarm を通じて直接リクエストに応じた処理を行い，結果を返す．この点が二つのモードの違いである．

5. 評価

本稿で実装したプロトタイプ実装について，複数の手法で評価を行った．

5.1 評価環境

評価を行った計算機の環境を表 3 及び 4 に示す．

表 3 のノードを 5 台用い，1 台を Gfarm のメタデータサーバである gfmd に割り当て，4 台をストレージサーバである gfsd に割り当てた．gfsd が割り当てられたノードにおいては，Apache HTTP Server もまた割り当てた．Gfarm の各デーモン間の通信は Infiniband FDR x4 上に構成された IPoIB によって行われる．クライアントには 4 で示すノードを 8 台用いた．クライアント側の計算機においてはネットワークインタフェースが Infiniband QDR x4 のため，サーバとの通信性能は Infiniband QDR x4 の性能に律速される．

表 3 評価を行ったノードの性能（サーバ側）

CPU	Intel(R) Xeon(R) CPU E5-2695 v2 2.40GHz (12 コア 12 スレッド) x2
メモリ	64GB
OS (Kernel)	CentOS 6.5 (2.6.32-431.5.1)
ネットワーク	Infiniband FDR 4X
ストレージ	146GB (6Gbps SAS, 15,000rpm) (1GB NVRAM)
Apache	2.4.9 (prefork)
Gfarm	2.5.8.7
PostgresQL	9.3.4

表 4 評価を行ったノードの性能（クライアント側）

CPU	Intel(R) Xeon(TM) E5620 CPU 2.40 GHz (4 コア 8 スレッド) x2
メモリ	24GB
OS (Kernel)	CentOS 6.5 (2.6.32-431.5.1)
ネットワーク	Infiniband QDR 4X

5.2 アクセス性能評価

この評価では，WebGfarm が単位時間に処理できるリクエスト数について評価を行う．WebGfarm は，HTTP のサービスとして動作するため，多数のクライアントから同時にリクエストが発行されることが想定される．この評価では，クライアントとなるプロセスからファイルを作成・読み書き・削除のリクエストをそれぞれ同時に複数生成した場合の性能について評価を行う．今回の評価においては，作成や読み書きを行うファイルの数はプロセス毎に 1,000 個であり，各ファイルへ読み書きを行うデータサイズは 4KB となっている．また，ファイルはすべて Gfarm 上の同一ディレクトリ上に作成した．WebGfarm のモードはプロクシモードとし，クライアントプロセス数を 1 から 32 まで変化させ評価を行った．

比較対象として WebGfarm と同様のインタフェースを提供する Apache モジュール mod_localfs を実装した．このモジュールではローカルのファイルシステムに対して WebGfarm と同様の操作を行う．これを gfarm2fs と組み合わせることで, WebGfarm と同様に HTTP 経由で Gfarm 上のファイルにアクセスすることが可能である．gfarm2fs を経由した場合においては FUSE などのオーバヘッドや制約があり，これらが性能にどのような影響を及ぼすか比較する．

mod_webgfarm の評価結果を図 3 に，mod_localfs/gfarm2fs の評価結果を図 4 にそれぞれ

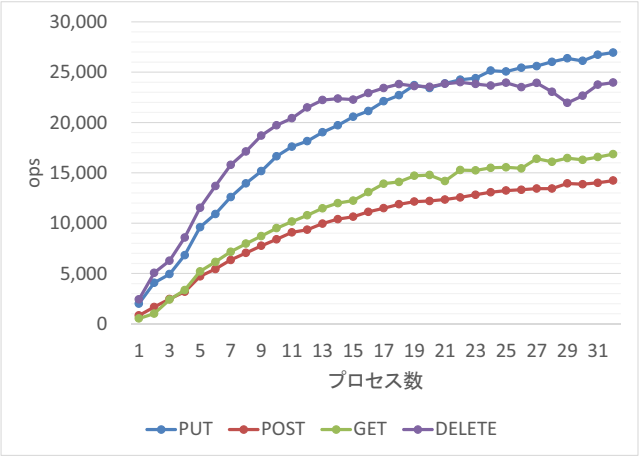

図 3 プロセス数と性能の関係 (mod_webgfarm)

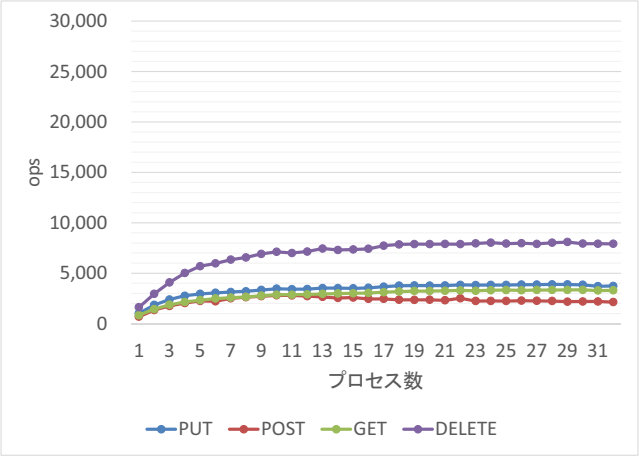

図 4 プロセス数と性能の関係 (mod_localfs/gfarm2fs)

示す．これらの図において，縦軸は 1 秒間に処理できたファイル数を示しており，数値が高い方が高い性能であることを示している．また，横軸はプロセス数を示している．凡例で PUT と示されているものが作成性能の結果であり，同様に POST が書き込み性能，GET が読み込み性能，DELETE が削除性能をそれぞれ示している．

これらの結果を比較すると，mod_webgfarm に比べ mod_localfs/gfarm2fs は少ないプロセス数で性能限界に達していることが示される．今回の評価においては，同一ディレクトリ内の複数のファイルを対象に操作を行ったが，FUSE を利用している gfarm2fs ではその場合にディレクトリエントリをロックしたためと考えられる．

5.3 ダイレクトモード、プロクシモードの性能評価

先のアクセス性能の評価においては，WebGfarm のモードとしてプロクシモードを用いた．今回の WebGfarm においては，動作モードとして他にもダイレクトアクセスモードがある．これらのモードを変えた場合に，どのような性能変化が見られるか先のアクセス性能の評価と同様の評価を行った．評価は 8 つのクライアントプロセスからリ

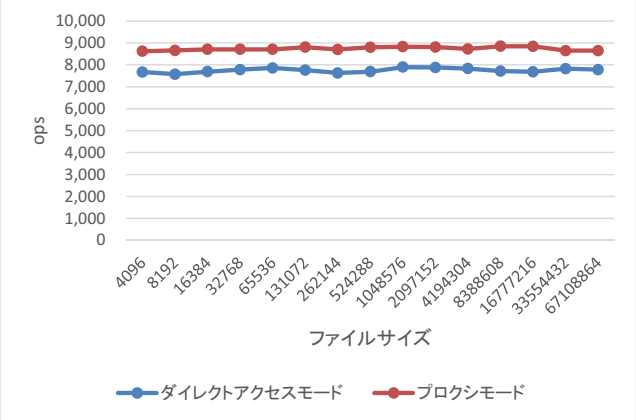

図 5 モードとファイルサイズの関係 (PUT)

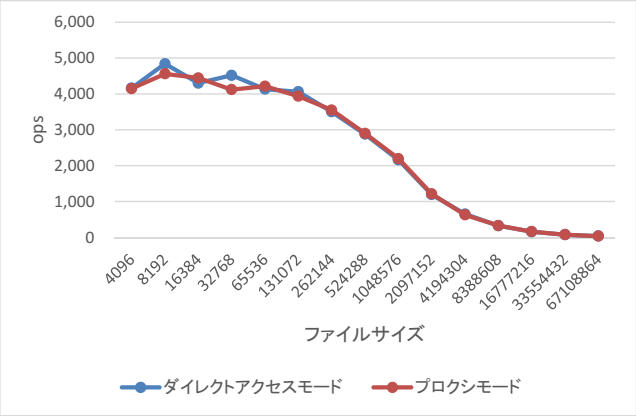

図 6 モードとファイルサイズの関係 (POST)

クエストを行い，それぞれのプロセスから 50 個のファイルを同一ディレクトリに作成した．また，読み書きのファイルサイズは 4KB から 64MB まで倍々に変化させて評価を行った．

モード差による性能変化の評価結果を図 5 から図 8 に示す．図 5 はファイル作成性能を示しており，図 6 は書き込み性能，図 7 は読み込み性能，図 8 は削除性能をそれぞれ示している．これらの図において縦軸はすべて一秒間に処理できたファイル数を示す ops(operations per second) であり，数値が高い方が高い性能であることを示している．横軸はファイルサイズであり，4KB から 64MB まで倍々に増えている．

ファイル作成性能の評価結果を示した図 5 においては，どちらのモードもファイルサイズの変化による性能変化は認められない．これはファイルを作成する操作においてはデータの書き込みも読み込みもないためである．ダイレクトアクセスモードとプロクシモードを比較すると，すべてのファイルサイズにおいてプロクシモードが約 10〜15%ほど性能が高くなっている．これは第 4 章において述べたように，ファイルの作成時においてもリダイレクトのためのスケジューリング処理が行われており，このオーバヘッドによる性能劣化であると考えられる．

© 2014 Information Processing Society of Japan

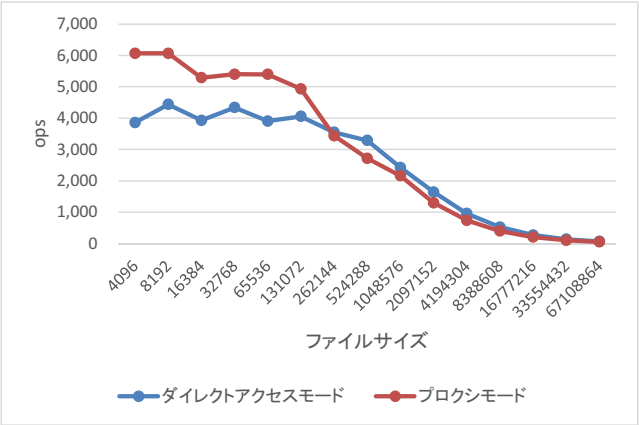

図 7 モードとファイルサイズの関係 (GET)

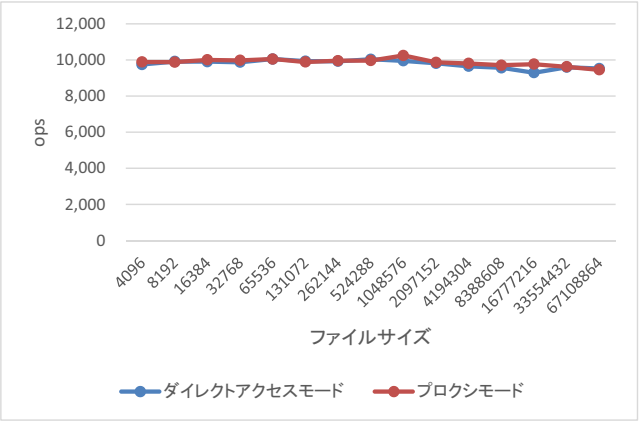

図 8 モードとファイルサイズの関係 (DELETE)

書き込み性能の評価結果を示した図 6 においては，どちらのモードもファイルサイズの増加に合わせて一秒間に処理したファイル数が減少している．これはファイルサイズが増えているためである．また，二つのモードを比較すると，性能差はほぼ見受けられない．この原因は今回のベンチマークがあらかじめ作成した 0 バイトのファイルに対して 1 度だけデータを書き込むからである．Gfarm においては，0 バイトのファイルが作成された段階ではそのファイルのデータを保存するストレージノードがまだ決定されない．ファイルへデータを書き込む際にストレージノードが決定されるが，今回の評価環境では httpd が動作している計算機上で Gfarm のストレージノードデーモンが動作している．このような場合においては Gfarm はなるべくローカルのストレージに書き込むため，ダイレクトアクセスモードにおいてもリダイレクトされず，どちらのモードでもほぼ同等の性能が示されたものと考えられる．

読み込み性能の評価結果を示した図 7 においても，両方のモードにおいてファイルサイズの増加に合わせて一秒間に処理したファイル数が減少している．また，二つのモードを比較するとファイルサイズが比較的小さい場合においてはプロクシモードの性能が高く，ファイルサイズが大きい場合においてはダイレクトアクセスモードの性能が高い．例えば，ファイルサイズが 4KB バイトの場合においてはプロクシモードはダイレクトアクセスモードの 157%の性能を示している．一方で，ファイルサイズが 64MB の場合においてはプロクシモードはダイレクトアクセスモードの 78%の性能にとどまっている．ファイル作成の評価結果においても述べたように，ダイレクトアクセスモードではデータローカリティを考慮したスケジューリングによるオーバヘッドがある．また，ダイレクトアクセスモードではリダイレクトによって HTTP のリクエストが一つ増える．Gfarm においては，libgfarm と gfsd 間のデータ転送が行われる．ダイレクトアクセスモードの場合は libgfarm と gfsd 間の通信は同一ホスト上にそれぞれのプロセスがあるためネットワークを超えない．しかしながら，プロクシモードの場合においては別のホストにデータがある場合もあり，その際はネットワーク上にデータが流れるオーバヘッドがある．これら二つのオーバヘッドの大きさの違いによって性能の高いモードが決定される．今回の場合においては，スケジューリングのオーバヘッドと HTTP のリクエストがひとつ増えることによるオーバヘッドはほぼ一定であるのに対し，ファイルをネットワーク上を転送させるオーバヘッドはファイルサイズに依存する．これらの結果，ファイルサイズが大きい場合においてはダイレクトアクセスモードの方が高い性能を示したと考えられる．

削除性能の評価結果を示した図 8 においてもファイル作成の評価結果と同様に，どちらのモードもファイルサイズの変化による性能変化は認められない．これはファイルの削除の操作においてはデータの書き込みも読み込みもないためである．また，二つのモードの性能差もほぼない．ファイルの削除においてはメタデータ操作のみとなることから，スケジューリングやそれによるリダイレクトは発生しない．これが二つのモードの性能差がほぼない要因と考えられる．

6. まとめ

本稿では，分散ファイルシステム Gfarm において HTTP で Gfarm 上のファイルにアクセスすることの出来るインタフェース WebGfarm を設計・実装を行い，初期評価としてアクセス性能評価と二つのモードにおける性能差について評価を行った．

評価では，アクセス性能評価において WebGfarm がプロセス数の増加に応じて高い性能を示すことを確認した．また，二つのモードにおける性能差については，GET 時においてファイルサイズが小さい場合ではプロクシモードがダイレクトアクセスモードの 157%の性能を示した一方で，大きなファイルの場合においては 78%の性能にとどまるなど，ファイルサイズに性能が依存することが示された．

今後の展望としては，以下のことが挙げられる．

(1) ファイルサイズに応じたモードの切り替え

ふたつのモード差の評価結果において，ファイルの読み込みの性能がファイルサイズに依存することが示された．読み込むファイルのサイズはあらかじめファイルシステム上にデータがあるため，この値を用いてモードを切り替える機構を用意した場合に性能が向上すると推測される．

(2) ファイルシステムのステータスの処理
本研究で実装を行ったプロトタイプはファイルやディレクトリ操作についてのみ可能である．Gfarm ではファイルシステムのステータスについても情報を提供する機能がある．この機能についても HTTP を経由してアクセスするインタフェースがあることが望ましい．

(3) 他の分散ファイルシステムへの応用
今回ターゲットとした分散ファイルシステムは Gfarm であったが，今回の WebGfarm を実装するにあたり用いた Gfarm の機能は第 4 章に示したとおりである．ここで用いたような関数に相当する機能が他の分散ファイルシステムに備わっていれば，WebGfarm と同等のプログラムを実装可能である．筆者らの研究チームは現在次世代分散ファイルシステムを設計している．この次世代分散ファイルシステムにおいても今回用いた libgfarm の機能を備えることで，同等のインタフェースを提供することが可能となる．

謝辞 本研究の一部は，JST CREST「ポストペタスケールデータインテンシブサイエンスのためのシステムソフトウェア」および，JST CREST「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」による．

参考文献

[1] Tatebe, O., Hiraga, K. and Soda, N.: Gfarm Grid File System, *New Generation Comput.*, Vol. 28, No. 3, pp. 257–275 (2010).
[2] FUSE: Filesystem in Userspace, http://fuse.sourceforge.net/.
[3] Kimura, H. and Tatebe, O.: MPI-IO/Gfarm: An Optimized Implementation of MPI-IO for the Gfarm File System, *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID '11, Washington, DC, USA, IEEE Computer Society, pp. 610–611 (online), DOI: 10.1109/CCGrid.2011.82 (2011).
[4] Mikami, S., Ohta, K. and Tatebe, O.: Using the Gfarm File System As a POSIX Compatible Storage Platform for Hadoop MapReduce Applications, *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, GRID '11, Washington, DC, USA, IEEE Computer Society, pp. 181–189 (online), DOI: 10.1109/Grid.2011.31 (2011).
[5] Shvachko, K., Kuang, H., Radia, S. and Chansler, R.: The Hadoop Distributed File System, *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, Washington, DC, USA, IEEE Computer Society, pp. 1–10 (online), DOI: 10.1109/MSST.2010.5496972 (2010).
[6] Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D. E. and Maltzahn, C.: Ceph: a scalable, high-performance distributed file system, *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, Berkeley, CA, USA, USENIX Association, pp. 307–320 (2006).
[7] Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures, PhD Thesis (2000).
[8] OpenStack Swift, http://docs.openstack.org/developer/swift/.
[9] WebHDFS REST API, http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/WebHDFS.html.
[10] HttpFS - Hadoop HDFS over HTTP, http://hadoop.apache.org/docs/r2.4.0/hadoop-hdfs-httpfs/index.html.
[11] The Apache HTTP Server, https://httpd.apache.org/.

© 2014 Information Processing Society of Japan