2007－HPC－113 (2)
2007／12／7

広域分散ファイルシステムGfarm v2の実装と評価

建部修見† 曽田哲之††

Gfarm v2 ファイルシステムは，小規模環境から大規模 PC クラスタ，広域分散環境までスケールすることを目指した広域分散ファイルシステムである．コモディティの利用，ファイル容量の動的増減，ファイル複製による高信頼性，分散アクセスによる高性能化に特徴がある．さらに，ファイルアフィニティを利用することにより，スケーラブルで効率的な分散データ処理も可能となる．本論文では，Gfarm v2 の実装について述べると共に性能評価を行う．

Implementation and Evaluation of Gfarm v2 Global Distributed File System

Osamu Tatebe† and Noriyuki Soda††

Gfarm v2 file system is a global distributed file system aiming to scale from a small workgroup to a large-scale enterprise, a high-end PC cluster, and a cluster of clusters in global environment. It is a commodity-based system characterized by flexible capacity change, high reliability by file replicas, high performance by distributed accesses. Moreover, it affords efficient and scalable parallel and distributed data computing exploiting file affinity. This paper discusses the implementation of Gfarm v2 file system, and shows preliminary performance results.

1. はじめに

コモディティPCとコモディティネットワークの高速化により，大規模なPCクラスタの構築が容易となり，大容量，高性能のファイルシステムの要求が出てきた．高エネルギー物理学，天文学，生物学などの科学技術分野だけではなく，ウェブサーバ，メールサーバなどの一般利用においても必要とされている．これら，安価に構築したPCクラスタに対するファイルシステムは，SANなどの利用は不適等であり，コモディティで構成されるものが望まれる．

本研究では，コモディティによる大容量，高性能なファイルシステムGfarm v2の実装と性能評価を行う．

2. 関連研究

大規模PCクラスタで利用される，大容量，高性能なファイルシステムとしていくつかの関連研究をあげる．Lustreファイルシステム[4]は，複数のOSTと呼ばれるディスクを束ねたファイルシステムであり，ストライピングするOSTの個数を指定し利用することができる．Gfarmと異なり，OSTはディスクを提供するだけであり，OST群と計算ノード間は高速なネットワークで接続する必要がある．また，ファイル複製を持たないため，耐故障性のためにはSANで複数パスを準備する必要がある．

PVFS2[5]はクラスタノードを利用した並列ストライピングファイルシステムである．単一の並列アプリケーションがMPI-IOを利用する場合の最適化はなされているが，本研究で対象としているような複数のアプリケーションからの同時アクセスに対しては，アクセス性能をスケーラブルにするのが難しい．

Googleファイルシステム[6]はコモディティ利用を前提としたファイルシステムある．故障に備え，各ファイルはデフォルトで3つのファイル複製をもつ．ただし，特定APIでアクセスをする必要があり，汎用的なファイルシステムではない．また，ソフトウェアが公開されていないため利用することができない．

Gfarmファイルシステムは，計算ノードのローカルファイルシステムを束ねるという点で，上記の関連システムと異なっている．そのため，ファイルシステムのための専用ノードやディスクアレイ，およびその間の高速ネットワークが必要なくなるというだけではなく，ローカルI/Oを活かしたスケーラブルなI/Oを活用できるという利点を持っている．

† 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
E-mail: tatebe@cs.tsukuba.ac.jp
†† (株) SRA
Software Research Associates, Inc.
E-mail: soda@sra.co.jp

3. Gfarm v2 の設計

Gfarm v2 の設計に関しては 9) でも一部議論されているが，Gfarm v1 の実装との差分という形での議論であったため，ここでは全体的な設計をまとめる．

3.1 検討項目と設計方針

コモディティを利用し，PC やクラスタノードなどのローカルファイルシステムを束ねることにより，ファイルシステムを構成する．束ねられた PC やクラスタノードをファイルシステムノードという．特別な装置や構成を仮定しないことにより，追加のハードウェアコストなしに直ちにファイルシステムの導入を可能とする．

ファイルシステム内部でファイル複製を管理し，ファイル複製を異なるローカルファイルシステムに保持する．これにより，ディスクや CPU，ネットワークの障害に対する耐故障性を実現する．アクセスが集中するファイルは，異なるファイル複製を参照することにより，アクセス集中による性能低下を防ぐ．また，ネットワーク的に近いファイル複製を参照することにより，アクセス遅延を緩和する．

現存するローカルファイルシステムの指定されたディレクトリ以下を束ねることにより，分散ファイルシステムを構成する．新規パーティションを要求することなく，現存するファイルシステムの領域をそのまま使えるようにすることにより，導入の準備が簡単になるだけではなく，ローカルファイルシステムを選ばない，ローカルファイルシステムの性能向上の恩恵を被ることができる，という長所を持っている．

Grid Datafarm アーキテクチャ[8] におけるスケーラブルなファイル入出力性能を実現するため，ローカルファイルシステム優先，ネットワーク的に近くて低負荷なファイルシステムノード優先のファイルシステムノードの選択を行う．さらに，バッチスケジューラと連携し，ファイルの格納場所でプログラムを実行するファイル・アフィニティ・スケジューリングを行うことより，さらなるスケーラビリティとファイル入出力性能の向上をはかる．

Gfarm v1 の実装で問題となっていた，以下の問題を解決する．

- ファイルシステムのメタデータに対する，ファイルシステムのディレクトリベースのアクセス制御
- 実際に格納されているファイルの保護
- メタデータアクセスの高速化によるファイル操作時間の短縮
- ファイルやディレクトリの移動，更新日時などのファイル属性の変更をメタデータの更新だけですませられるようにする
- ユーザ，グループの管理
- 複数のプロセスが同一ファイルをオープンした場合のファイル複製間の一貫性の保証
- ファイルシステムノードに不必要なファイルが残らないようにする

3.2 ファイルシステム・メタデータ

ファイルシステムのディレクトリ情報，ファイル情報，ファイル格納場所など，ファイルシステムの管理のために必要な情報をファイルシステム・メタデータと呼ぶ．Gfarm ファイルシステムでは，ファイルシステム・メタデータをファイルの格納から分離し，独立に管理する．ファイル，ディレクトリの管理を基本的に i ノード番号で行うことにより，ファイルやディレクトリの移動に対し，実際に格納されているファイルの移動は伴わず，メタデータだけの更新ですませられるようにする．

3.3 ユーザ，グループ管理

Gfarm ファイルシステムは，複数の管理ドメインをまたぐ広域のファイルシステムとしての機能を提供するため，Gfarm ファイルシステムに対するグローバルなユーザ名，グループ名の管理が必要である．Gfarm v2 では，その管理をメタデータサーバで行う．

このとき，利用ユーザのグローバルなユーザ名へのマッピングをどのように行うかが問題となる．この問題は，個人をどのように認証するかにより，解決策が変わってくる．GSI (Grid Security Infrastrcture) 認証[3] の場合，ユーザ証明書の Subject DN (Distinguished Name) がユーザを特定する固有の名前と考えられ，Subject DN と Gfarm ファイルシステムのグローバルユーザ名のマッピングをメタデータで管理すればよい．共有秘密鍵認証の場合は，ユーザを一意に特定するための ID が特に存在しないため，それぞれのローカルサイトにおいて，ローカルユーザ名と Gfarm のグローバルユーザ名のマッピングを管理する．

3.4 ファイルシステムノードのモニタリング

束ねられたファイルシステムノードの状況のモニタリングは，現在利用可能なファイルシステムノードの把握，複数あるファイル複製の選択のため，ファイル作成における適切なファイルシステムノードの選択のために必要である．

モニタリングでは，ファイルシステムノードが利用可能であるか，および利用可能な場合は，CPU 負荷，ストレージ容量，空き容量などの情報が求められる．Gfarm v2 では，ファイルシステムノードとメタデータサーバの間には接続があるため，この接続を利用してメタデータサーバが定期的にファイルシステムノードの状態を取得する．メタデータサーバがファイルシステムノードの状態を定期的に取得し，ファイルシステムノードの利用可能性，および各種スケジューリングのための情報として利用する．

4. Gfarm v2 の実装

4.1 ソフトウェア構成と概要

Gfarm v2 は，メタデータサーバ (gfmd)，I/O サーバ (gfsd)，Gfarm ライブラリ (libgfarm)，Gfarm コマンド，gfarm2fs で構成される．

gfmd はファイルシステム・メタデータの管理を行い，メタデータアクセス時に，ファイルシステムのディレクトリ単位の細かなアクセス制御を行う．メタデータは，基本的にはメタデータサーバのメモリに保持し，メタデータアクセスに対する高速性を確保する．

gfsd は，ファイルシステムノード上で動作する I/O サーバであり，束ねられるローカルファイルシステムへアクセスするために用いられる．各 gfsd と gfmd の間には TCP の接続がなされ，ファイル open 時に指定されたファイルディスクリプタを i ノード番号に変換する，ファイル close 時にファイル情報を更新するなどの処理に利用される．また，gfmd との接続は，gfmd によるファイルシステムノードの情報の定期的なモニタリング，実ファイルの消去のためにも利用される．

クライアントからのアクセスは，gfarm2fs および Gfarm コマンドによりなされる．gfarm2fs は，FUSE[1] を利用したユーザレベルで Gfarm ファイルシステムをマウントするためのプログラムである．各ファイルシステムノード，クライアントノードで gfarm2fs を利用して，Gfarm ファイルシステムをマウントすることにより，既存のプログラムから Gfarm ファイルシステムをアクセスすることが可能となる．

Gfarm コマンドは，ファイル複製作成，ファイル複製の格納場所の参照など，既存のファイルシステムに対するインターフェースには含まれない機能を提供する．

4.2 gfmd - メタデータサーバ

gfmd は，ファイルシステム・メタデータ，ファイルの open 状態，ファイルシステムノードの状態，グローバルなユーザ名とグループ名の管理を行う．

ファイルシステム・メタデータの管理では，ファイルの i ノード，ディレクトリエントリ，ファイル複製の格納場所の管理を行う．i ノードには，i ノード番号，世代番号，リンク数，ファイルサイズ，アクセス制御情報，ユーザ名，グループ名，アクセス時間，修正時間，状態変化時間が格納されている．

ファイルの open 状態の管理では，プロセスとファイルディスクリプタの管理がなされる．

ファイルシステムノードの管理では，ファイルシステムノードが利用可能かどうかの情報と，利用可能な場合は，1 分，5 分，15 分平均の CPU 負荷，ディスクの利用容量，全容量，および情報取得時間が保持される．

グローバルなユーザ名の管理では，ユーザ名，ホームディレクトリ，リアルネーム，ユーザ証明書の Subject DN で管理される．グループに関しては，グループ名の管理と所属しているユーザ名の管理がなされる．

メタデータに対するアクセスに関しては，ディレクトリベースのアクセス制御を行う．メタデータアクセスを高速化するため，メタデータは，なるべくメモリに保持する．これにより，クライアント，I/O サーバからの要求の応答に対して disk アクセスが必要なくなり，高速となる．

遠隔からのアクセスに関しては，メタデータサーバに対するアクセス回数が問題となる．例えば，アメリカ西海岸から日本にアクセスする場合，往復のネットワーク遅延は 100 msec 程もあるため，ファイルアクセス操作の遅延を減らすためには，アクセス回数を減らすことが特に重要となる．そのために，Gfarm v2 では，NFSv4[7] にも導入されているコンパウンドプロシージャをベースに，エラーの場合にエラーの種類に応じた処理が行えるよう拡張したコンパウンドプロシージャを用いている．

突然のメタデータサーバの故障や電源断に備え，メタデータはバックエンドのデータベースにも保持する．メタデータの更新時には，バックエンドデータベースも更新を行う必要があるが，このデータベースの更新はバックエンドで行う．すなわち，メタデータの変更をメモリ上のキューにため，データベースへの書込スレッドが別スレッドでデータベースに書き込む．これにより，キューが一杯にならない限り，メタデータ操作は disk アクセス待ちとはならない．

メタデータの分散管理，冗長管理に関してはまだ実装されていない．ただし，メタデータの耐故障性，フェイルオーバに関しては，バックエンドのデータベースを冗長にすること，およびメタデータサーバの切替えにより対応することが可能である．

4.3 gfsd - I/O サーバ

gfsd は，ローカルファイルシステムに対するアクセス機能を提供する．ファイル open 時には，クライアントプロセスから，ファイルディスクリプタ番号を受け取り，そのファイルディスクリプタ番号を元に，メタデータサーバに i ノード番号を問い合わせ，該当ファイルのアクセスを行う．

ファイル close 時には，メタデータサーバで管理される該当ファイルの i ノードの内容を更新する．このとき，クライアントプロセスからの明示的なファイル close 前に，クライアントプロセスからのネットワーク接続が切断された場合，クライアントプロセスの abort と判断して，i ノードの内容を更新する．これにより，クライアントプロセスの突然の終了時にも，実ファイルの情報と一貫したメタデータの更新が可能となる．

gfsd は，起動時にメタデータサーバに接続し，利用

可能であることを伝える．この接続は，gfmd によるハートビート，ファイルシステムノードの状態取得，実ファイルの消去要求などに利用される．

4.4 Gfarm ライブラリ

Gfarm ライブラリは，Gfarm ファイルシステムをアクセスするための Gfarm API の実装であり，gfmd, gfsd からも共通して利用されるライブラリを含んでいる．

Gfarm API の詳細は 8) および 2) にあるが，ファイルの open, close, read, write, seek, stat, rename, unlink といった基本操作，およびファイル複製関係の API からなる．

初期化では，gfmd に接続し，実行プロセスの登録を行う．ファイルの open 時には，gfmd に問い合わせ，ファイルディスクリプタ番号を取得する．ファイルアクセス時には，gfsd の選択を行う必要があるが，その際，gfmd から取得するファイルシステムノード情報とクライアントからのネットワーク距離を考慮する．

クライアントからのネットワーク距離の計測の詳細はここでは割愛するが，基本的には，ファイルシステムノードのネットワークグループによるグループ化と，ネットワークグループごとのネットワーク往復遅延 (RTT) の計測からなる．

gfsd の選択が終了すると，選択した gfsd に対し，ファイルの open を行い，その後，read, write, seek などのファイルアクセスを要求する．ファイルの close 時には，選択した gfsd および gfmd に対して close の要求を行うことになる．

gfmd へのアクセスはファイルの open 時，close 時だけであり，read, write などのファイルアクセス時は gfsd を直接アクセスする．そのため，ファイルアクセスのバンド幅に対しては，gfmd へアクセスするオーバヘッドはかからない．

ファイルが更新された場合，ファイル複製間で一貫性を保持する必要がある．Gfarm v2 では，メタデータサーバにより各プロセスのファイル open の状態を管理し，Close-to-open コンシステンシを基本に考える．Close-to-open コンシステンシは、AFS などで用いられているセマンティクスであり，更新されたファイルの内容が他のプロセスから参照されるのは，そのファイルをクローズした後というものである．

4.5 Gfarm コマンドと gfarm2fs

Gfarm コマンドは，既存のファイルシステムに対するインターフェースには含まれない複製処理，Gfarm ファイルシステムの管理，gfarm2fs を利用しないで Gfarm ファイルシステムを直接参照する機能などを提供する．

gfarm2fs は，FUSE を利用し，Gfarm ファイルシステムをマウントするためのプログラムである．

	筑波大	AIST	SDSC
#nodes	14	8	3
RTT [msec]	0.202	0.787	119

表 1 各拠点のノード数と各拠点のノードからメタデータサーバまでのネットワークの往復遅延時間 (msec).

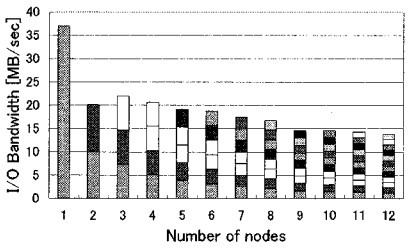

図 1 NFS における I/O バンド幅．クライアントのノード数を増やすと全体の I/O バンド幅は下がっていく．

5. 性能評価

Gfarm v2 の実装の性能評価にあたり，複数クライアントからの並列 I/O におけるバンド幅，およびファイル操作における所要時間を，NFS と比較することにより評価を行う．

5.1 性能評価環境

筑波大，産総研 (AIST)，サンディエゴスーパコンピュータセンター (SDSC) の三拠点のクラスタを利用して Gfarm ファイルシステムを構成した．メタデータサーバは筑波大で立ち上げ，筑波大の 14 ノード，AIST の 8 ノード，SDSC の 3 ノードをファイルシステムノード兼クライアントノードとした．各拠点のノード数と，各拠点のノードから筑波大のメタデータサーバまでのネットワークの往復遅延時間 (RTT) を表 1 に示す．筑波大内の同一 LAN であれは RTT は約 0.2 msec，AIST と筑波大の間は約 0.8 msec，SDSC と筑波大の間は約 120 msec である．

比較のための NFS サーバは，筑波大のクラスタと同一 LAN のノードを利用し，筑波大の 14 ノードをクライアントノードとした．

全てのノードにおいて，OS は Linux 2.6 を利用し，Gfarm で束ねているローカルファイルシステムは ext3 である．NFS サーバも Linux 2.6 であり，HDD8 台の RAID6 に ext3 のファイルシステムを構成し，async モードで export している．

5.2 複数クライアントからの I/O バンド幅

複数のクライアントが異なる 1 GByte のファイルを同時に読み込むときの I/O バンド幅の性能評価を行う．まず，比較のために，筑波大の同一 LAN 内のクライアントから NFS におかれたファイルを読み込むときの性能を図 1 に示す．

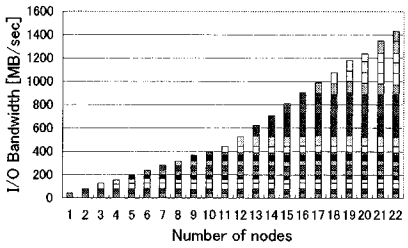

図 2 Gfarm ファイルシステムにおける I/O バンド幅．クライアントのノード数を増やすと全体の I/O バンド幅はスケーラブルに向上している．

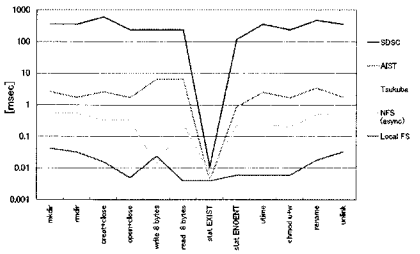

図 3 ローカルファイルシステム，NFS および Gfarm ファイルシステムにおける各ファイル操作の所要時間．同一 LAN の環境であれば，NFS と Gfarm ファイルシステムはほぼ同程度であり，Gfarm ファイルシステムに対する遠隔からのアクセスは 2 ～ 4 RTT 程である．

クライアント数が 1 のときは，37.0 MB/sec であったが，クライアント数が 2 ～ 4 のときは，全体の I/O バンド幅は 20 ～ 22 MB/sec となり，さらにクライアント数を増やすと徐々に下がっていく．クライアント数が 12 のときの全体の I/O バンド幅は 14 MB/sec となった．この結果より，複数のクライアントが異なる 1 GByte のファイルを読み込む場合，全体の I/O バンド幅はクライアント数が増えると下がっていくことが分かる．

一方で，Gfarm ファイルシステムにおける I/O バンド幅の性能を図 2 に示す．Gfarm ファイルシステムの場合は，筑波大の同一 LAN 内のノードだけではなく，筑波大，AIST，SDSC の三拠点のノードを利用している．クライアント数が 1 ～ 11 では，筑波大のクライアントノードからの同時アクセス，クライアント数が 12 ～ 19 のときは，筑波大と AIST のクライアントノードからの同時アクセス，クライアント数が 20 ～ 22 のときは，筑波大，AIST，SDSC の三拠点のクライアントノードからの同時アクセスとなっている．

筑波大，AIST，SDSC と日米に分散したクライアントからのアクセスであるにもかかわらず，クライアント数を増やすと，増やした分 I/O バンド幅が増加している．全体の I/O バンド幅はクライアント数を増やせば増やす程増加し，クライアント数が 22 のとき，1433 MB/sec を達成している．この結果より，複数のクライアントが異なる 1 GByte のファイルを読み込む場合，全体の I/O バンド幅はクライアント数が増えると増えた分だけ増加し，スケーラブルな性能向上が得られていることが分かる．

5.3 ファイル操作の所要時間

Gfarm v2 における性能改善の大きな目標はファイル操作の所要時間の短縮であり，ファイル操作の各項目に関して性能評価を行う．性能評価にあたり，Gfarm のソフトウェアディストリビューション[2] に含まれる fsysbench を利用した．fsysbench は，ディレクトリ作成 (mkdir)，存在するファイル，しないファイルの stat，ファイルの作成 (creat+close)，ファイルの open (open+close)，utime，chmod，8 バイト書込 (write 8 byte)，8 バイト読込 (read 8 byte)，rename，ファイルの消去 (unlink)，ディレクトリの消去 (rmdir) などのファイル操作における所要時間を計測するベンチマークプログラムであり，ディレクトリ数およびそれぞれのディレクトリ内のファイル数を指定し，一操作あたりの平均所要時間を計測する．

図 3 にローカルファイルシステム，NFS および Gfarm ファイルシステムにおける fsysbench の結果を示す．ローカルファイルシステムは，筑波大のノードの Gfarm ファイルシステムで束ねているローカルファイルシステム (計算ノードのローカルファイルシステム) を直接利用したものである．

ローカルフィルシステムを直接アクセスする場合の所要時間は，ディレクトリの作成，消去，ファイルの消去に 0.032 ～ 0.042 msec，8 バイト書込に 0.024 msec，ファイル作成，rename に 0.016 ～ 0.018 msec，ファイルの open，8 バイト読込，stat，chmod，utime では 0.004 ～ 0.006 msec であった．

NFS では，ディレクトリの作成，消去，ファイルの消去に 0.52 ～ 0.55 msec，rename に 0.49 msec，ファイルの作成，open に 0.33 msec，8 バイト読込，存在しないファイルの stat，utime，chmod に 0.20 ～ 0.25 msec，8 バイト書込に 0.013 msec，存在するファイルの stat に 0.007 msec であった．

メタデータサーバと同一 LAN 上の筑波大のクライアントノードからの Gfarm ファイルシステムに対するアクセスでは，rename に 0.70 msec，ディレクトリの作成，消去，ファイルの作成，消去，8 バイト書込，utime に 0.43 ～ 0.59 msec，8 バイト読込，chmod に 0.36 ～ 0.37 msec，ファイルの open に 0.28 msec，存在しないファイルの stat に 0.17 msec，存在するファイルの stat に 0.004 msec であった．

NFS と比較すると，8 バイト書込で 0.4 msec 程遅

くなっているが，これは NFS を async モードで export したためである．他の操作では，ファイルの作成，utime，rename で 0.2 msec 程，ディレクトリの作成，8 バイト読込，chmod で 0.1 msec 程遅くなっている一方で，ディレクトリの消去，ファイルの open，存在しないファイルの stat，ファイルの消去では 0.1 msec 程速くなっている．これらにより，同一 LAN におけるファイル操作の所要時間は NFS と Gfarm ファイルシステムでほぼ変わらないといえる．なお，存在するファイルの stat は，全ての場合でキャッシュされており，stat システムコールは実際には発行されていない．

AIST のクライアントノードからは，8 バイト書込，読込で 6.45 ～ 6.50 msec，rename で 3.48 msec，ディレクトリの作成，ファイルの作成，utime で 2.50 ～ 2.64 msec，ディレクトリの消去，ファイルの open，chmod，unlink で 1.69 ～ 1.75 msec，存在しないファイルの stat で 0.85 msec，存在するファイルの stat で 0.004 msec であった．

筑波大の同一 LAN 内のクライアントからのそれぞれの操作にかかる時間は，AIST と筑波大間のネットワークの RTT よりも短いため，メタデータサーバとの通信が大きな影響を及ぼすと考えられる．

ファイルシステム・メタデータの操作だけで完了するものでは，rename で RTT のほぼ 4 倍，ディレクトリの作成，utime で RTT のほぼ 3 倍，ディレクトリの消去，chmod で RTT のほぼ 2 倍，存在しないファイルの stat でほぼ RTT の時間がかかっている．また，実際のファイルアクセスを伴う 8 バイトの読込，書込では若干所要時間が延びているが，ファイルの作成で RTT のほぼ 3 倍，ファイルの open，unlink で RTT のほぼ 2 倍の時間となっていることが分かる．

SDSC のクライアントノードからは，ファイルの作成に 596 msec，rename に 477 msec，ディレクトリの作成，消去，utime，unlink に 358 msec，ファイルの open，8 バイト書込，読込，chmod に 238 ～ 240 msec，存在しないファイルの stat に 119 msec，存在するファイルの stat に 0.01 msec であった．

ファイルシステム・メタデータの操作だけで完了するものは，AIST の場合と同様であるが，ディレクトリの消去は，AIST では RTT の 2 倍であったが，SDSC では 3 倍であった．本件は，今後より詳細な評価が必要である．また，実ファイルへのアクセスを伴うものでは，ファイルの作成で RTT のほぼ 5 倍となっているが，ファイルの消去では RTT のほぼ 3 倍，ファイルの open，8 バイトの読込，書込では RTT のほぼ 2 倍の時間となっていることが分かる．

これらの評価により，同一 LAN 内であれば，ほぼ NFS と同程度の時間，遠隔のクライアントからのアクセスであれば，ほぼ RTT の 2 倍から 4 倍 (SDSC におけるファイル作成は 5 倍) の時間となっており，メタデータアクセスの高速化とメタデータサーバとの通信回数の削減ができていることが分かった．

6. ま と め

コモディティを利用した分散ファイルシステムの Gfarm v2 の実装についてまとめ，性能評価を行った．日米の 3 拠点の PC クラスタからの並列ファイルアクセス性能の評価では，クライアント数にスケーラブルな I/O バンド幅を得ることができ，22 クライアントで 1,433 MB/sec の性能を達成した．また，ファイル操作の所要時間に関する評価では，同一 LAN では NFS とほぼ同等，遠隔からのアクセスでは RTT の 2 ～ 4 倍の時間であった．

謝　　辞

本研究における性能評価のため，PC クラスタを利用させていただいた産業技術総合研究所，およびサンディエゴスーパコンピュータセンターの関係各諸氏に感謝いたします．本研究の一部は，情報爆発時代に向けた新しい IT 基盤技術の研究，文部科学省科学研究費補助金「特定領域研究」(課題番号 19024009) による．

参 考 文 献

1) *FUSE: Filesystem in Userspace*. http://fuse.sourceforge.net/.
2) *Grid Datafarm*. http://datafarm.apgrid.org/.
3) *The Grid Security Infrastructure Working Group*. http://www.gridforum.org/security/gsi/index.html.
4) *Lustre*. http://www.lustre.org/.
5) *PVFS*. http://www.pvfs.org/.
6) Ghemawat, S., Gobioff, H. and Leung, S.-T.: The Google File System, *Proceedings of 19th ACM Symposium on Operating Systems Principles (SOSP-19)* (2003).
7) Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M. and Noveck, D.: *Network File System (NFS) version 4 Protocol* (2003). RFC 3530.
8) 建部 修見, 森田 洋平, 松岡 聡, 関口 智嗣, 曽田 哲之: ペタバイトスケールデータインテンシブコンピューティングのための Grid Datafarm アーキテクチャ, 情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム, Vol. 43, No. SIG 6 (HPS 5), pp. 184–195 (2002).
9) 建部 修見, 曽田 哲之, 関口 智嗣: 広域仮想ファイルシステム Gfarm v2 の設計と実装, 情報処理学会研究報告 2004-HPC-99, pp. 145–150 (2004).