ペタスケール広域分散データ解析のための Grid Datafarm アーキテクチャ☆

建部修見[†1] 森田洋平[†2] 松岡聡[†3]
関口智嗣[†1] 曽田哲之[†4]

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 (Gfarm) アーキテクチャの設計と実装を行っている．Gfarm は，グリッド上の数万台規模の PC クラスタのローカルディスクで構成されるデータ並列ファイルシステムにより，オンラインでペタバイト規模の大容量，ローカル I/O バンド幅を利用したスケーラブル I/O バンド幅，スケーラブル並列処理を提供する．初期性能評価では Presto III Athlon クラスタ 16 ノードを利用し，Gfarm 並列 I/O による読み込み，書き込みにおいて，それぞれ 352.2MB/sec，375.4MB/sec を達成した．また，Gfarm 並列ファイル複製では 100Mbps Fast Ethernet において 8 並列で 89.4MB/sec を達成した．

Grid Datafarm Architecture for Global Petascale Data Intensive Computing

Osamu Tatebe,[†1] Youhei Morita,[†2] Satoshi Matsuoka,[†3]
Satoshi Sekiguchi[†1] and Noriyuki Soda[†4]

Grid Datafarm (Gfarm) architecture is designed for global Petascale data intensive computing, that provides a global data parallel filesystem with online Petascale storage, scalable I/O bandwidth and scalable parallel processing exploiting local I/O of a cluster of clusters with ten-thousands of nodes on the Grid. Preliminary performance evaluation shows scalable disk I/O and network bandwidth on 16 nodes of the Presto III Athlon cluster. Gfarm parallel I/O write and read achieve 352.2MB/sec. and 375.4MB/sec., respectively, using 16 nodes. Gfarm parallel file copy achieves 89.4MB/sec. with eight parallel streams on 100Mbps Fast Ethernet.

1. はじめに

高エネルギー物理学，天文学，地球惑星物理学，人ゲノムなどの大規模データ解析を必要とする研究分野では，ハイパフォーマンスコンピューティング，データインテンシブコンピューティング，ネットワーク技術が不可欠となってきた．一つの例は，2006 年より開始される予定になっているスイス CERN の Large Hadron Collider (LHC) 実験プロジェクトである．LHC 実験には 4 つの測定器，実験グループがあり，それらの測定器は毎年ペタバイトオーダの測定データを生成する．それぞれの実験には数十ヶ国規模，数千人規模の素粒子物理学者が参加し，実験データの解析において協力および競争することになる．MONARC プロジェクト [16] では，世界規模の階層的な地域センタの計算モデルについての研究が行われた．この地域センタモデルでは，0 層センタは CERN におかれ，1 層センタはヨーロッパ，アメリカ，アジアなど，2 層センタは各国，3 層センタはそれぞれの大学，研究所におかれる．地域センタ間にはギガビット級の高速ネットワークが整備され，(1) それら広域高速ネットワークの帯域を利用する技術，(2) 地域センタ間におけるファイルの複製とその管理，(3) ペタバイトスケール大容量データ処理技術，に関する研究開発が特に必要とされている．

これらペタバイトスケールのデータインテンシブコンピューティングにとり，グリッド技術，クラスタ技術，ネットワーク技術は実装のための鍵となっている．大容量データを扱うグリッド環境は特に Data Grid と呼ばれ，米国および欧州では EU DataGrid[1]，GriPhyN[5]，PPDG[10]，iVDGL[9]，GridPP[7] など大型プロジェクトが複数立ち上がり，Data Grid のためのグリッドサービスの研究開発，Data Grid 環境構築，

☆ http://datafarm.apgrid.org/

†1 産業技術総合研究所グリッド研究センター
National Institute of Advanced Industrial Science and Technology
E-mail: {o.tatebe,s.sekiguchi}@aist.go.jp
†2 高エネルギー加速器研究機構計算科学センター
High Energy Accelerator Research Organization
E-mail: youhei.morita@kek.jp
†3 東京工業大学学術国際情報センター
Tokyo Institute of Technology
E-mail: matsu@is.titech.ac.jp
†4 (株)SRA
Software Research Associates, Inc.
E-mail: soda@sra.co.jp

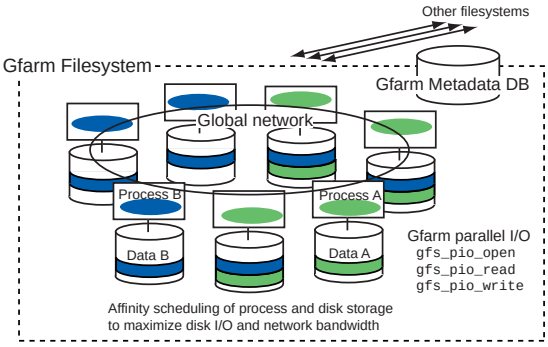

図 1 Gfarm コアアーキテクチャ構成図

標準化などが行われている．標準化に関しては，主に Global Grid Forum[2] で議論が行われ，ファイル転送では，FTP にグリッド認証 [6] と，並列ストリームによる広域高速転送の拡張を行った GridFTP[20] が提案されている．複製管理に関しては，Globus replica management[11] が多く利用されているが，いまだ研究開発段階というところである．

Grid Datafarm (Gfarm) は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上記問題を解決するためのアーキテクチャである．データインテンシブ処理において，典型的な場合では大容量データはレコードあるいはオブジェクトの集合であり，大部分の処理はそれら全オブジェクトに対する検索や解析などオブジェクトに対して独立した処理となる．Gfarm では，これら典型的であり，大部分を占める処理に対し，並列ストライピングファイルシステムの拡張と，計算とデータのアフィニティスケジュールを利用したデータ並列ファイルシステム (Gfarm ファイルシステム) を提供する．Gfarm ファイルシステムでは，ファイルはファイル断片に分割されて分散配置され，計算はそれぞれの断片が格納されているノードにスケジューリングされ，ローカル I/O を活用することにより，ペタバイトスケールのスケーラビリティを目指している．さらに，このファイルシステムを広域グリッド環境に広げることによって，上記 (1)，(2)，(3) の問題の解決を図っている．

2. Gfarm コアアーキテクチャ

図 1 に Gfarm コアアーキテクチャの構成図を示す．Gfarm はグリッド上の数千から数万ノードの PC クラスタのクラスタで構成され，そのローカルディスクを用い Gfarm ファイルシステムが構成される．

2.1 Gfarm ファイルシステム

Gfarm ファイルシステムは，数万ノードのディスクを利用して，ペタバイトを越えるファイルをスケーラブルなディスク I/O バンド幅で扱うことができる並列ファイルシステムであり，グリッドサービスとして提供される．ペタバイトスケールの大規模ファイルを扱うことができ，それら大規模ファイルは，物理的には断片に分割され，高速ネットワーク上に分散するディスクに分散配置される．分割は，ストライピングファイルシステムとは異なり，一定のブロックサイズによる分割だけではなく，それぞれの断片のサイズは自由に動的に決定することができる．また，分散されるノードに関する制限もなく，自由なノード，自由なノード数に任意のファイルを分散させることができる．

ファイルはファイル断片単位に複製を持つことができ，複製は故障時のバックアップ，参照時のバンド幅，遅延，負荷分散などによる選択のために利用される．

2.1.1 Gfarm ファイルとメタデータ

Gfarm ファイルシステムにおけるファイルは，Gfarm ファイルと呼ばれ `gfarm:/path/name` のような形の Gfarm ファイル名あるいは Gfarm URL で表される．この時，ファイル断片の分散，格納されているノード，複製などを意識することなくアクセスすることができる．Gfarm URL からそれらファイル断片の分散，格納場所，複製カタログなどの情報はファイルサイズ，プロテクション，アクセス/修正/変更時刻などのファイル状態情報といっしょに Gfarm メタデータデータベースにより管理される．

Gfarm メタデータにはチェックサム，生成ヒストリも含まれる．チェックサムはデータ化けなどのチェック，複製間の一貫性のチェックなどに利用され，生成ヒストリは，ノードやディスクの障害時におけるデータの再計算，データ生成プロセス解析などに利用される．

メタデータは，Gfarm ファイルシステムのファイル操作とともに更新され，一貫性が保証される．一般的には，ファイルのオープン時にメタデータが参照され，クローズ時に更新，一貫性のチェックが行われる．並列プロセスのうちのどれかのプロセスが正しくメタデータを更新せず，エラーや故障などで異常終了してしまった場合，そのメタデータは一貫性がとれないため，チェック時に消去される．

2.1.2 ローカル I/O を利用したスケーラブル並列 I/O

それぞれのノードでは *gfsd* と呼ばれる I/O デーモンが走り，I/O ノードとして並列にアクセスすることができる．gfsd は，リモートアクセス，アクセス制御だけではなく，ユーザ認証，プログラム実行，ファイル複製，ノードの資源モニタリングなども行う．

一般的に並列ファイルシステムは，複数 I/O ノードという並列性を利用して高バンド幅を得ているが，この並列性は，ネットワークバンド幅，ネットワークスイッチのバンド幅に押さえられてしまう．ペタバイトスケールのデータに対しては，GB/s であっても読み込むだけで 10 日以上かかってしまい，少なくとも TB/s 程度のバンド幅が要求される．そのため，このアプローチでは，ネットワークのコストが問題となる．

Gfarm ファイルシステムでは，それぞれのノードはI/O ノードだけではなく計算ノードでもあり，I/O ノードからデータをネットワークを利用して転送するだけではなく，それぞれのノードのローカル I/O を積極的に利用する．ローカル I/O はノード数に対しスケーラブルに増加するため，それぞれのノードがたとえ数十 MB/s のバンド幅であっても，数万ノードで TB/s 程度のバンド幅を達成することができる．

2.2 プロセスとファイルのアフィニティスケジューリングと Gfarm 並列 I/O

ローカル I/O を利用するために，Gfarm では特定のファイル断片に分割された Gfarm ファイルに関して並列プロセスをスケジューリングすることができる．この場合，そのファイルが分割されているノード数と同数の並列プロセスが生成され，それらのプロセスは可能な限りファイル断片が格納されているノードで実行される．さらに，Gfarm 並列 I/O API が提供され，そのスケジューリングに利用された Gfarm ファイルに関して，それぞれのプロセスが担当するファイル断片を対象とするファイルビューが提供される．また，このファイルビューは新しく生成する Gfarm ファイルに関しても適応することができる．

大規模ファイルに対し，この並列プロセススケジューリングと担当ファイル断片を対象とするファイルビューを利用することにより，それぞれのプロセスの大多数のファイルアクセスはローカル I/O を利用したスケーラブルなファイル I/O となる．

2.3 障害時におけるファイル復元

Gfarm ファイルシステムは，Gfarm システムにおいて実行されるプログラムも格納することができる．実行形式ファイルの場合，メタデータとして OS，CPU アーキテクチャなど実行プラットフォーム情報も格納される．プログラムの実行時には，ローカルディスクにそのプログラムの複製が存在しない場合，まず複製が作成され，その後実行される．

すべての Gfarm ファイルは基本的に write once である．ファイルの書き換えに関しては，ファイル名を変更して新規ファイルとするか，あるいは同じファイル名でもバージョニングを行う．これは，(1) 大規模データは頻繁に変更されるというよりは，read only あるいは write only のデータが多いということ，(2) 書き換えが無い場合は複製に関して一貫性維持の必要性がなくなること，(3) 書き換えが無い場合は生成ヒストリによりファイルの再生成が可能となること，などのためである．

Gfarm では，ファイル生成時に利用されたプログラムとすべての引数は生成ヒストリとしてメタデータに保存される．プログラムおよびすべての引数は Gfarm ファイルシステムに存在するため，ファイル生成時に利用したのと同じプログラム，同じ引数ファイルを利用することにより，Gfarm システムで二次的に生成されたファイルは再計算で復元することができる．また，Gfarm ファイルは消去されてもそのメタデータは消去されないため，消去された Gfarm ファイルが二次的に生成されたものであれば，再計算により復元することができる．このことにより，Gfarm ファイルシステムに一次的に登録されたファイルは複製により，二次的あるいはそれ以降に生成されたものは，複製あるいは再計算により，データを復元することができる．

2.4 広域における認証

Gfarm システムはグリッドサービスとして提供され，グリッドセキュリティインフラストラクチャ(GSI)[6] による安全な相互認証を利用することができる．GSI を利用しているため，それぞれの資源はシングルサインオンで利用することができる．

しかしながら，Gfarm システムの場合，並列プロセス，メタデータデータベース，gfsd などのシステム内部において実行時に認証が必要となり，プロセス数，ノード数が数千，数万となるとその認証オーバヘッドが実質的に大きくなってしまう．そのため，Gfarm ではいくつかの認証方式を利用し，安全なクラスタ内では共有秘密鍵による軽い認証から，グリッド上における GSI までを提供する．

2.5 アプリケーション例

ペタバイトスケールのデータインテンシブ処理では，データ位置の関してローカリティの高い処理が多く，しかも大半の処理を占める．LHC 実験においても，大半の処理は全イベントデータに対するイベントごとの処理であり，数 PB に及ぶデータのなかでも，数 MB のイベントデータに関する独立した処理となっている．それら，数 PB に及ぶ大規模データに関して並列プロセスのアフィニティスケジューリングを行うことにより，もっとも重たい処理に関してほとんどの処理を，ローカル I/O を利用したスケーラブルな処理とすることができる．

図 1 では，プロセス A はデータ A の分散されているノードでスケジューリングされている．プロセス B は，同様にデータ B の分散されているノードでスケジューリングされるが，データ B はデータが三分割され，それぞれのファイル断片が複製されているため，ノードの負荷状況などによりそのうち 3 ノードがスケジューリングされている．

また，これらに関しての典型的な例としては UNIX の`grep` 処理があげられる．`grep` 処理は，ファイルを行単位で読み込み，指定されたパターンを含む行を出力するものである．この場合，対象ファイルを行単位で適当なサイズにブロック分割しておき，`grep` 処理は，その対象ファイルに関してスケジューリングする．それぞれの並列プロセスは，多くの場合自ノードのローカルディスクに格納されていることが期待される，担当のファイル断片を読み込んで処理することになる．出力は，対象ファイルのファイルビューと同じファイル

ビューで書き出すことにより，同じく自ノードのローカルディスクに作成されることが期待される．しかしながら，あくまでプログラムは，Gfarm URL でファイルを参照，生成しており，単一システムイメージは保存されている．

3. Gfarm 並列 I/O API

Gfarm 並列 I/O API は，Gfarm ファイルシステムに対する単一システムイメージによる並列ファイルアクセスと，さらにローカル I/O を利用したスケーラブルなバンド幅を実現するための並列ファイルアクセスを提供する．

Gfarm ファイルは，複数のインデックス付けされたファイル断片に分割され，複数のディスクに格納される．ファイルアクセスは，ファイル全体，特定のファイル断片など，ファイルビューによってアクセス範囲が制限される．

Gfarm では，ファイル全体に対するアクセスでも，並列ファイルシステムとして，高バンド幅を得ることができるが，特定のファイル断片に対する並列アクセスを用いることにより，ローカル I/O のスケーラビリティを利用したアクセスが可能となる．Gfarm では，特定の Gfarm ファイルのファイル断片の総数および格納ノードを並列プロセスのスケジューリングに利用することができ，その場合，それぞれのプロセスは担当するファイル断片を持ち，この担当するファイル断片をファイルビューとすることにより，ローカル I/O のスケーラビリティを利用したアクセスとなる．このファイルビューは，新規ファイルの作成時にも利用することができるため，読み込みだけではなく，書き込み時においてもスケーラブルなバンド幅を得ることができる．

Gfarm 並列 I/O API は UNIX のファイル操作システムコールのほとんどすべてが提供される．ほとんどすべての API は完了メッセージの定数アドレスを返し，エラー，エラーメッセージのチェックがポータブルに簡単に可能となっている．API のより詳細については，4)，18)，22) を参照されたいが，以下の API はそのうち主要な部分，および変更部分となっている．

3.1 ファイル操作

3.1.1 ファイルのオープン，作成

```
char* gfs_pio_open(char *url, int flags,
    GFS_File *gf);
char* gfs_pio_create(char *url, int flags,
    mode_t mode, GFS_File *gf);
```

gfs_pio_open は Gfarm URL `url` で指定された Gfarm ファイルをオープンし，新たな Gfarm ファイルハンドル `gf` を返す．`flags` は以下のリストの bitwise-inclusive-OR で指定する．以下のファイルアクセスモードはどれか一つを必ず指定する必要がある．

`GFARM_FILE_RDONLY`　読み込みモード
`GFARM_FILE_WRONLY`　書き込みモード
`GFARM_FILE_RDWR`　読み書きモード

以下は，任意のコンビネーションが可能であるが，`GFARM_FILE_REPLICATE` と `GFARM_FILE_NOT_REPLICATE` は同時に指定することができない．また，これらは単に効率的実行のための実行時のヒントであり，必ずその通りに実行されるとは限らない．

`GFARM_FILE_SEQUENTIAL`　シーケンシャルアクセスの指定
`GFARM_FILE_REPLICATE`　必要であればファイルを複製する．
`GFARM_FILE_NOT_REPLICATE`　遠隔アクセスであってもファイルを複製しない．

gfs_pio_create は Gfarm URL `url` で指定される Gfarm ファイルをアクセスモード `mode` で作成し，新たな Gfarm ファイルハンドル `gf` を返す．`mode` はアクセス許可を指定するが，アクセス許可はプロセスの `umask` でマスクされる．

gfs_pio_open と gfs_pio_create は，並列プロセスで発行した場合，それぞれのプロセスでは独立の個別ファイルポインタを持つ．

3.1.2 ファイルビュー

並列プロセスをスケジューリングするのに利用された Gfarm ファイル，および新たに作成された Gfarm ファイルでは，以下の API によりそれぞれのプロセスが担当するファイル断片を対象とするファイルビューに設定することができる．

```
char* gfs_pio_set_view_local(GFS_File gf,
    int flags);
```

gfs_pio_set_view_local は，Gfarm ファイルハンドル `gf` で示されるファイルのファイルビューを，それぞれのプロセスが担当するファイル断片に対するビューに変更する．`flags` の指定は gfs_pio_open のヒントの指定と同じである．

並列プロセスをスケジューリングするのに利用された Gfarm ファイルをこのファイルビューで参照する場合は，それぞれのプロセスがローカルディスクに格納されていることが期待されるファイル断片を，それぞれ独立にアクセスすることとなる．また，新たに作成される Gfarm ファイルの場合は，ローカルディスクに空きが十分あれば，それぞれのファイル断片をローカルディスクに作成する．新たに作成する場合，ファイル断片の順序は，プロセスランクと同じ順序となる．

以下の API は，ファイル断片を明示的に扱うためのファイルビューである．

```
char* gfs_pio_set_view_index(GFS_File gf,
    int nfrags, int index, char *host,
    int flags);
```

gfs_pio_set_view_index は，Gfarm ファイルハンドル `gf` で示されるファイルのファイルビューを，インデッ

クス index で指定されるファイル断片とする．新たに生成するファイルの場合は，ファイル断片の総数 nfrags および作成するノード名 host を指定する．既にある Gfarm ファイルを参照する場合は，nfrags および host には GFARM_FILE_DONTCARE を指定することができる．この場合，host に関して複製が存在する場合も含め，それらの値はメタデータデータベースから得られることとなる．flags の指定は gfs_pio_open のヒントの指定と同じである．

3.1.3 ファイルのクローズ

```
char* gfs_pio_close(GFS_File gf);
```

gfs_pio_close は Gfarm ファイルハンドル gf をクローズし，Gfarm メタデータデータベースを更新あるいは確認する．

3.2 ファイルアクセス

Gfarm 並列 I/O API では，ファイルアクセスに対しブロッキング，非集合的，個別ファイルポインタの操作を提供する．

```
char* gfs_pio_read(GFS_File gf, void *buf,
                   int size, int *nread);
```

gfs_pio_read はファイルハンドル gf で参照される Gfarm ファイル断片から size バイト読み，buf に格納し，nread に実際に読んだバイト数を返す．

```
char* gfs_pio_write(GFS_File gf, void *buf,
                    int size, int *nwrite);
```

gfs_pio_write は，ファイルハンドル gf で参照される Gfarm ファイル断片に buf から size バイト書き込み，nwrite に実際に書き込んだバイト数を返す．

3.3 従来のオブジェクトコートおよび商用アプリケーションのポーティングのためのシステムコールトラップ

商用データベースなどのように，ソースコードが利用可能ではない，あるいは修正することができないような従来のオブジェクトコード，あるいは商用アプリケーションに関しては，ファイル I/O に関するシステムコールをトラップすることにより，Gfarm ファイルシステムの利用，および Gfarm による並列プロセススケジューリングなどの並列化を可能とする．この場合，open，write，close などのシステムコールがトラップされ，数千，数万のファイルが自動的に一つの Gfarm URL としてグループ化される．Gfarm URL としてグループ化されたファイルは，Gfarm ファイルシステム管理の対象となり，並列プロセススケジューリング，動的負荷分散と耐故障性などのための自動複製生成のために利用される．

トラップされたシステムコールは，それぞれのプロセスが担当するファイル断片に関するファイルビューとして実行される．open や creat などのシステムコールは，パス名が Gfarm URL かどうかチェックし，Gfarm URL の場合は gfs_set_view_local により，それぞれのプロセスのファイル断片に関するファイルビューとする．そのときのファイルディスクリプタは Gfarm URL のものとして登録され，それ以降に実行される read や write システムコールでは，そのファイルディスクリプタが Gfarm URL をオープンしたものかどうかを調べ実行される．

```
write_test(char *fn, void *buf, int size)
{
    GFS_File gf;
    gfs_pio_create(fn, GFS_FILE_WRONLY,
                   mode, &gf);
    gfs_pio_set_view_local(gf, lflag);
    gfs_pio_write(gf, buf, size, &np);
    gfs_pio_close(gf);
}
```

図 2 Gfarm API を利用した I/O バンド幅測定のプログラム

4. Gfarm コマンド

Gfarm コマンドは Gfarm ファイルシステムを操作するシェルレベルのコマンドであり，ほとんどの UNIX ファイル操作コマンドおよび Gfarm 管理コマンドからなる．以下はそのうち主要なものだけであるが，より詳細については，4)，18)，22) を参照されたい．

gfls，gfmkdir，gfrmdir などのディレクトリに関する操作は，Garm メタデータデータベースのファイルメタデータを操作し，それぞれファイル，ディレクトリのリスト，ディレクトリの作成，消去を行う．gfrm，gfchmod，gfchown などのファイルに関する操作は，ファイルメタデータと Gfarm ファイルシステムの Gfarm ファイル断片を操作し，それぞれファイルの消去，モード，所有者の変更を行う．これらの引数となるファイル名は Gfarm URL で指定される．

5. 予備実験評価

Gfarm の予備評価を東工大の Presto III Athlon クラスタを用いて行った．Presto III のそれぞれのノードの CPU は dual AMD Athlon MP 1.2GHz であり，100Mbps Fast Ethernet および Myrinet 2000 で現在 128 ノード，256 プロセッサが接続されている．Gfarm メタデータベースサーバは LDAP サーバを用い，Fast Ethernet で接続される dual Pentium III 500MHz のノードを利用した．それぞれの OS は Linux 2.4 である．

5.1 I/O バンド幅

Gfarm の I/O バンド幅の性能評価にあたり，Gfarm 並列 I/O API を利用し，ファイル断片をそれぞれのノードのローカルディスクにおいて測定した．このとき，それぞれ書き込みに関する部分のプログラムは**図 2** のようになる． ファイル名は Gfarm URL で指定され，書き込み時は gfs_pio_create および gfs_pio_set_view_local により（十分に空きがあれば）それぞれのノードのローカルディスクに新しいファイル

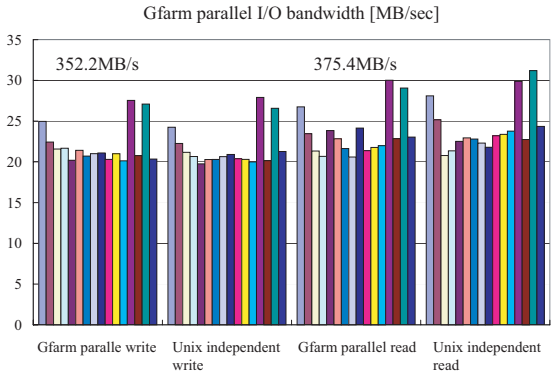

図 3 Gfarm 並列 I/O 性能

断片を並列に作成し，gfs_pio_write によりそのファイル断片に並列に書き込み，gfs_pio_close によりクローズする．このクローズ時に Gfarm メタデータデータベースのメタデータは更新される．読み込みに関しては，まず `fn` として指定される Gfarm URL によりプロセススケジューリングを行い，gfs_pio_open により Gfarm メタデータデータベースのメタデータが参照され，gfs_pio_set_view_local によりそれぞれのノードに割り当てられているファイル断片を並列にオープンする．一般的には，それぞれのノードに割り当てられているファイル断片は，それぞれのノードのローカルディスクにあるとは限らず，この場合，遠隔ファイルアクセスあるいは複製作成となる．が，性能評価にあたっては，それぞれのファイル断片はそれぞれのローカルディスクにあるようにスケジューリングした．また，gfs_pio_set_view_local ではそれぞれのノードに複数ファイル断片が割り当てられている場合，それらを連続的にアクセスするが，性能評価では，それぞれのノードは単一のファイル断片を割り当てた．その後，gfs_pio_read によりそのファイル断片を並列に読み込み，gfs_pio_close によりクローズする．読み込み時に md5 チェックサムが計算され，クローズ時にメタデータとして登録されている md5 チェックサムと比較され，データ化け，複製間の一貫性などのチェックが行われる．

図 2 では，全バッファ領域を一度に書き込んでいるが，実際の測定プログラムでは同じ 64K バッファに対して書き込み，あるいは読み込みを行っている．また，図 2 ではエラーチェックは省いているが，実際はそれぞれの Gfarm API の呼び出しについて必要である．

図 3 に 16 ノードを利用し，16GB データを読み書きした時の Gfarm 並列 I/O のバンド幅を示す．それぞれのノードではメモリバッファの影響を軽減し，ディスク I/O 性能を測定するため，主記憶容量の 768MB を越える 1GB のデータの読み書きとなっている．また，これらの性能は図 2 で示されるプログラムのうち，オープンからクローズまでの時間を元にしており，

表 1 Fast Ethernet におけるファイル複製のバンド幅（8 並列）

TCP	TCP + disk I/O	Gfarm `gfrep`
92.5	90.2	89.4

[MB/sec.]

Gfarm メタデータデータベースへのアクセス，チェックサム計算のオーバヘッドなどが含まれている．

Gfarm 並列書き込みでは 16GB のデータに対し，合わせて 352.2MB/s のバンド幅を達成している．この場合，それぞれのノードの平均は 22.0MB/s である．それぞれのノードで独立にファイルをオープン，書き込み，クローズした場合の性能を Unix independent write として示している．総計では 346.8MB/s となり，Gfarm 並列書き込みより若干性能が下がっているが，この差はマルチユーザモードによる測定誤差，ディスクブロックの割り当ての問題と考えられる．

Gfarm 並列読み込みでは総計で 375.4MB/s のバンド幅を達成している．この場合，それぞれのノードの平均は 23.4MB/s である．一方で，それぞれのノードで独立にファイルをオープン，読み込み，クローズした場合の性能は，総計で 386.3MB/s となっている．

これらの結果より，16 ノードまではスケールしていること，メタデータアクセスオーバヘッドおよびチェックサム計算のオーバヘッドは隠れていることが分かる．

5.2 並列ファイル複製

表 1 に並列ファイル複製のバンド幅を示す．並列ファイル複製にあたり，8 ノードにファイル断片が分散された 8GB のファイルを，ネットワーク越しに別の 8 ノードに `gfrep` コマンドを用い複製した．`gfrep` コマンドは，書き込み先の複数の gfsd に接続し，それぞれの gfsd に対しファイル断片のコピーを起動させるため，本評価においては 8 ストリームを利用した並列ファイル複製ということになる．

Gfarm 並列ファイル複製では，並列に転送することにより 89.4MB/s を達成している．このバンド幅は Fast Ethernet のバンド幅 100Mbps および単体のディスク I/O のバンド幅 22 ∼ 23MB/s を大幅に越える性能である．

`TCP` は netperf2.1pl3 を利用した 8 本の TCP Stream の性能であり，`TCP + disk I/O` はネットワーク転送に加え，ディスクの読み書きを伴った場合の性能である．一方で，`gfrep` は `gfrep` の起動時間，メタデータ参照，複数の gfsd への接続，コピー要求，メタデータ更新など全てのオーバヘッドを含んだ場合の性能である．それらの性能を比較すると，`gfrep` のバンド幅のオーバヘッドは 1%以下であることが分かる．

6. 関 連 研 究

並列ファイルアクセスのための標準 API として，MPI-IO[15] が定められている．しかしながら，MPI-

IO で定められているファイルビューは MPI ユーザ定義データ型の配列の延長であり，Gfarm で提供している動的にファイルブロックのサイズが変化するブロック分割ともいえる，担当ファイル断片に対するファイルビューのようなものは定義することができない．また，Gfarm では，Gfarm ファイルを並列プロセスのスケジューリングのために利用することができ，これらによりローカル I/O によるスケーラビリティを利用している．

Linux クラスタのローカルディスクを利用した並列ファイルシステムとして PVFS[12] が開発されている．PVFS は striping によりファイルを分割し分散させ，それらの間は TCP を利用して通信する．Native PVFS API だけではなく UNIX/POSIX I/O API および MPI-IO をサポートしており，さらに mount して利用することもできる．一方，Gfarm は，ファイルの格納されている PC で処理を実行することによりローカル I/O バンド幅を最大限に活用しているところ，故障に頑強であること，また広域および数千台規模以上を考慮していることが異っている．

商用の並列ファイルシステムは IBM SP の PIOFS, GPFS[14] などがあるが，これらはそれぞれのマシンでしか動作せず，また広域，異機種では動作しない．

HPSS[8] は分散階層型ストーレジシステムであり，IP ネットワークで接続されたストライピングディスクキャッシュ，並列ムーバなどによる並列 I/O により高バンド幅を達成している．並列 FTP，MPI-IO，DFS が主な利用手段となる．しかしながら，データを処理するためには HPSS からデータを計算機に移動させて，データ処理し，また書き戻すという操作が必要となるため，HPSS と計算機の間のネットワークのバンド幅がボトルネックとなってしまう．Gfarm ではデータをなるべく移動させずそれぞれのデータが格納されているノードで並列に処理することにより，このネットワークバンド幅のボトルネックを解消している．

分散ファイルシステムは，NFS, AFS, Coda をはじめ xFS[13], GFS[17] など様々開発されている．これらのシステムは基本的に，多くのクライアントからの共有ディスクに対する分散アクセスを可能とするものである．しかしながらこれらの方式は，複数からの読み込みに関しては，ディスクキャッシュなどでアクセス効率をあげることができるが，並列アプリケーションが典型的に必要とする，複数からの書き込みに対しバンド幅を確保することができない．

グリッド上のファイル転送，遠隔ファイルアクセスのためのシステムとしては GridFTP[20], Legion I/O[21], Kangaroo[19] など様々なシステムが提案されている．GridFTP は FTP に GSI[6]，および広域ネットワークにおいて高い転送バンド幅を実現するための適応的並列ストリームなどの拡張を行ったものである．Kangaroo では，ローカルディスクをディスクキャッシュとして利用し広域における遅延を隠すと同時に，広域環境で起こりがちな一時的ネットワーク不通などの回復可能なエラーをユーザに対し隠している．Gfarm では，広域ネットワーク上のクラスタ間におけるファイル転送において並列ストリームにより高いバンド幅を提供するだけではなく，大容量のデータではなく計算プログラムを転送し，それぞれのデータが格納されているノードで実行することによりネットワークバンド幅を越えるスケーラブルなディスクバンド幅の実現を目指している．

Globus replica management[11] はグリッドにおいて複製されたファイルのメタデータを管理し，アクセス時に最適な複製を選択するためのサポートをするものである．メタデータ操作のためのレプリカカタログ API が提供されているが，この場合，メタデータと実際のファイルの一貫性の保証はその API を利用するアプリケーションにまかせられる．Gfarm ではメタデータはファイルシステムのメタデータとして，ファイル操作に基づき一貫して管理され，それらを直接操作する API は提供されないため，これらの問題はない．この問題に対して，Globus replica management では，レプリカカタログ API と GridFTP を利用したファイルコピーとレプリカカタログ更新操作を一緒にして一貫性を保つレプリカ管理 API が設計され，Globus Toolkit 2 リリース [3] において利用可能となっている．

7. まとめと今後の課題

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では，スケーラブルな I/O バンド幅，スケーラブル並列データ処理の実現が重要となる．Gfarm では広域における数千台規模の PC クラスタのローカル I/O を積極的に利用することにより，ペタバイトスケールを越えるスケーラビリティを目指している．数千，数万ノードのローカル I/O を利用するために，それらにつながるローカルディスクにより並列ファイルシステムを構築し，さらにそれらに分散したファイルのファイル断片に対するデータ並列処理のための Gfarm 並列 I/O API と，ファイル断片と処理プログラムのアフィニティスケジューリングを利用する．ファイルシステムとして単一システムイメージを与えつつ，大多数のアクセスをローカルディスクのアクセスとすることにより，数千，数万ノードにおけるスケーラビリティを目指している．

広域における数千，数万台規模のクラスタのクラスタに対して，Gfarm ファイルシステムを構築するためには，ノードの故障，ディスクの故障，ネットワークの故障などに対応する必要がある．Gfarm では，それら故障に対応するため，ファイルは複製することができ，それら複製はメタデータで一貫して管理される．また別の手段として，メータデータとして管理される

生成ヒストリを利用して再実行することにより，ファイルを復元することもできる．ファイルの複製は，耐故障性のためだけではなく，並列データ処理のアフィニティスケジューリングにおける負荷分散のためにも作成される．

予備性能評価では，Presto III Athlon クラスタ 16 ノードにおいてスケーラブルな I/O バンド幅を実現した．並列書き込みでは 352.2MB/s，並列読み込みでは 375.4MB/s を達成し，また並列ファイル複製では 100Mbps Fast Ethernet において 8 ストリームを用い 89.4MB/s を達成した．ローカル I/O を利用した並列書き込み，読み込みでは Gfarm によるメタデータのアクセスなどのオーバヘッドはほぼ隠れており，また並列ファイル複製に関してもプロセス起動，並列ファイル転送起動，メタデータアクセスなどを含むオーバヘッドによるバンド幅の低下は 1%以下であった．

現在はプロトタイプシステムとして必要最小限の機能が動作しており，今後，更にノードを増やした場合の性能評価，広域のクラスタのクラスタによる性能評価およびアプリケーションによる性能評価が必要である．また，自動複製作成などを含むスケジューリング，耐故障性など完全なシステムの開発も必要であるが，アーキテクチャ的には数 TB/sec の I/O バンド幅も実現可能と思われ，ペタスケールデータインテンシブコンピューティングにおいては必須の技術と考えられる．

Gfarm は，CERN LHC 実験プロジェクトに同期し，2005 年までにはペタスケールのオンラインディスクシステムの構築を目指しているが，それ以外のバイオインフォマティックス，天文学，地球惑星物理学などのデータインテンシブコンピューティングへの適応も検討している．

謝　　辞

本研究を遂行するにあたり貴重なご助言，ご討論いただいた産業技術総合研究所，高エネルギー加速器研究開発機構，東京工業大学，東京大学の Gfarm プロジェクトメンバ諸氏，産業技術総合研究所大蒔和仁情報処理研究部門長，グリッド研究センターのメンバ諸氏に感謝致します．

参 考 文 献

1) *EU DataGrid Project*. http://www.eu-datagrid.org/.
2) *Global Grid Forum*. http://www.gridforum.org/.
3) *Globus Toolkit 2 release*. http://www.globus.org/gt2/.
4) *Grid Datafarm*. http://datafarm.apgrid.org/.
5) *Grid Physics Network*. http://www.griphyn.org/.
6) *The Grid Security Infrastructure Working Group*. http://www.gridforum.org/security/gsi/index.html.
7) *The GridPP Project*. http://www.gridpp.ac.uk/.
8) *HPSS: High Performance Storage System*. http://www.sdsc.edu/hpss/.
9) *International Virtual Data Grid Laboratory*. http://www.ivdgl.org/.
10) *Particle Physics Data Grid*. http://www.ppdg.net/.
11) Allcock, B., Bester, J., Bresnahan, J., Chervenak, A. L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S.: Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing, *Proceedings of IEEE Mass Storage Conference* (2001).
12) Carns, P. H., Ligon III, W. B., Ross, R. B. and Thakur, R.: PVFS: A Parallel File System for Linux Clusters, *Proceedings of the 4th Annual Linux Showcase and Conference*, pp. 317–327 (2000). http://www.parl.clemson.edu/pvfs/.
13) Anderson, T. E., Dahlin, M. D., Neefe, J. M., Patterson, D. A., Roselli, D. S. and Wang, R. Y.: Serverless network file systems, *Proceedings of the Fifteenth ACM Symposium of Operating Systems Principles*, pp. 109–126 (1995).
14) Corbett, P. F., Feitelson, D. G., Prost, J.-P., Almasi, G. S., Baylor, S. J., Bolmarcich, A. S., Hsu, Y., Satran, J., Snir, M., Colao, R., Herr, B., Kavaky, J., Morgan, T. R. and Zlotek, A.: Parallel file systems for the IBM SP computers, *IBM Systems Journal*, Vol. 34, No. 2, pp. 222–248 (1995).
15) Message Passing Interface Forum: *MPI-2: Extensions to the Message-Passing Interface* (1997). http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html.
16) MONARC Collaboration: Models of Networkd Analysis at Regional Centres for LHC Experiments: Phase 2 Report, Technical Report CERN/LCB-001, CERN (2000). http://www.cern.ch/MONARC/.
17) Soltis, S. R., Ruwart, T. M. and O'Keefe, M. T.: The Global File System, *Proceedings of the Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies* (1996). http://www.globalfilesystem.org/.
18) Tatebe, O., Morita, Y., Matsuoka, S., Soda, N., Sato, H., Tanaka, Y., Sekiguchi, S., Watase, Y., Imori, M. and Kobayashi, T.: Grid Data Farm for Petascale Data Intensive Computing, Technical Report ETL-TR2001-4, Electrotechnical Laboratory (2001). http://datafarm.apgrid.org/pdf/gfarm-ETL-TR2001-4.pdf.
19) Thain, D., Basney, J., Son, S.-C. and Livny, M.: The Kangaroo Approach to Data Movement on the Grid, *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing*, pp. 325–333 (2001).
20) The Globus Project: *GridFTP: Universal Data Transfer for the Grid*. http://www.globus.org/datagrid/deliverables/C2WPdraft3.pdf.
21) White, B. S., Grimshaw, A. S. and Nguyen-Tuong, A.: Grid-Based File Access: The Legion I/O Model, *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing*, pp. 165–173 (2000).
22) 建部 修見, 森田 洋平, 松岡 聡, 関口 智嗣, 曽田 哲之: 広域大規模データ解析のための Grid Datafarm アーキテクチャ, 情報処理学会研究報告 2001-HPC-87, pp. 177–182 (2001).