ハイパフォーマンス
コンピューティング 87−31
（2001. 7. 27）

広域大規模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 アーキテクチャの設計と実装を行っている．Grid Datafarm は，PC クラスタのローカルディスクを利用した広域データ並列ファイルシステムを提供し，オンラインでペタバイト規模の大容量と，ローカル I/O バンド幅を利用したスケーラブルな I/O バンド幅が特徴である．Gfarm 並列 I/O API および Gfarm コマンドにより，単一システムイメージの操作を可能とする．ファイルの複製，ヒストリによる再生成などにより，自動的な耐故障性，負荷分散も目指している．

Grid Datafarm Architecture for Petascale Data Intensive Computing

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

Design of Grid Datafarm architecture for Petascale data intensive computing is described. Grid Datafarm provides global data parallel filesystems with online Petascale storage and scalable I/O bandwidth to exploit local disks of group of PC clusters on the Grid. Gfarm parallel I/O APIs and Gfarm commands provide a single system image for the filesystem. Automatic management of fault-tolerance and load balancing is also an important issue, which is done by file duplication and re-computation using a command history.

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 LHC(Large Hadron Collider) 実験プロジェクトである．LHC 実験には 4 つの観測器，実験グループがあり，それらの観測器は毎年ペタバイトオーダの観測データを生成する．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 [2] 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．広域に分散するため，グリッド技術はこれらの実装のための鍵となっている．

Gfarm(Grid Datafarm) は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環境の構築のため，産業技術総合研究所 (AIST)，高エネルギー加速器研究機構 (KEK)，東京大学素粒子物理国際研究センター (ICEPP)，東京工業大学の共同研究で始まった．目標は，LHC の ATLAS 実験による数百テラバイトから数ペタバイト規模の実験データ解析環境の構築である．ICEPP と KEK は共同で日本に ATLAS 実験の 1 層地域センタを構築することになっている．想定しているハードウェアは，それぞれのノードがテラバイト級のローカルディスクを持つ数千台規模の PC クラスタである．CERN からやってくる 600Mbps ほどの実験データは並列に系統的にそれらのディスクに格納され，それぞれの PC で処理される．Gfarm で

---

* http://datafarm.apgrid.org/

†1 産業技術総合研究所
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
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

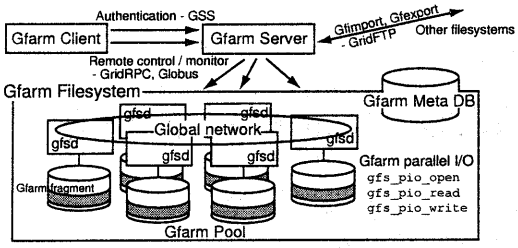

図 1 Gfarm アーキテクチャ

は，LHC 実験に加え，広くデータインテンシブコンピューティングのために以下の機能を提供する．

- ペタバイトスケールのファイルを扱うためのグローバル分散ファイルシステム
- 並列 I/O と並列処理
- 世界規模の認証とアクセス制御
- 数千ノード，広域の資源管理とスケジューリング
- 階層的データ共有と効率的アクセス
- プログラム共有と管理
- システムモニタリングと管理
- 耐故障性 / 動的再配置 / 動的データ復元，再計算

Gfarm の主要構成要素は，Gfarm クライアント，Gfarm サーバおよび Gfarm 並列 I/O を行う Gfarm 並列分散ファイルシステムである．Gfarm ファイルシステムは，数千ノードスケールの PC クラスタであり，それぞれのノードのローカルディスクを利用して，広域大規模ファイルシステムを実現する．ペタスケールのデータは，断片に分割されそれぞれのローカルディスクに分散配置され，Gfarm 並列 I/O ライブラリを用いそれぞれの PC により並列にアクセスされる．Gfarm ファイルシステムは，Gfarm メタデータベースにより管理され，Gfarm ファイルシステムデーモンにより，遠隔ファイルアクセス，実行プログラム複製，資源管理など必要な操作が行われる．

2. Gfarm コアアーキテクチャ

図 1 に Gfarm アーキテクチャを示す．Gfarm は大きく Gfarm クライアント，Gfarm サーバ，Gfarm プールからなり，それらを合わせて Gfarm システムと呼ぶ．Gfarm プールは広域の数千から数万ノードの PC 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．ペタバイトスケールの大規模ファイルは，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．それらファイルの断片の情報および論理パス名からの変換等のファイルシステムのメタ情報は，ファイルシステムメタデータベースが管理する．ファイル断片に対する I/O は Gfarm 並列 I/O API を用い，論理パス名によりアクセスされる．この論理パス名は Gfarm ファイル名あるいは Gfarm URL と呼ばれ，`gfarm:/path/name` のように表される．

Gfarm ファイルシステムは，ただのファイルシステムではなく，ペタバイトスケールのデータインテンシブ処理のために，並列 I/O，並列処理を前提として設計されている．基本的な実行イメージでは，プログラムはそれらファイル断片の割り当てられている PC で実行され，ファイルアクセスはローカルアクセスとなる．プログラムは，Gfarm 並列 I/O API を用い論理名である Gfarm ファイル名でファイルを参照，生成するが，実際はローカルなファイル断片を参照し，新しいローカルなファイル断片を作成し，新しい Gfarm ファイル名をメタデータとして登録する．この場合，ファイル I/O のバンド幅は，分散している PC の数にスケーラブルに増加するため，バンド幅的にはペタバイトスケールの大規模データ処理に対応することができると考えられる．

Gfarm ファイルシステムデーモン (gfsd) は Gfarm プールのそれぞれのノードで実行され，主にアクセス制御，リモートディスクのアクセスに利用される．gfsd では，軽量な GFS RPC を用い，低オーバヘッドのアクセスを可能にしている．gfsd は，ファイルアクセス以外にも，Gfarm サーバに登録されている実行形式の動的ローディング，ノードの資源のモニタ，制御も行う．

Gfarm ファイルシステムのメタデータは，論理 Gfarm ファイル名からそれぞれの断片の物理ファイル名へのマッピング，ファイルサイズ，プロテクション，アクセス/修正/変更時刻，チェックサムなどのファイル状態情報，ファイル複製カタログ，生成ヒストリなどで構成され，Gfarm メタデータベースで管理される．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，データ正当性の検証のための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，それぞれのファイル断片のクローズ時に登録され，すべての並列プロセスが終了したとき，その論理 Gfarm ファイル名のメタデータがチェックされる．もしどれかのプロセスがメタデータを正しく登録しないで，エラーなどで終了してしまった場合，そのメタデータは一貫性がとれていないため，チェック時にそのメタデータは消去される．

Gfarm サーバは，Gfarm プール，Gfarm ファイルシステムを効率的に利用するよう拡張したネットワーク可用サーバ (network-enabled server)[6] である．Gfarm サーバはまず，GSS(Generic Security Service)[5] あるいは IPsec を用い，Gfarm クライアントを相互に認証する．認証は Gfarm システムに対して一度だけのシングルサインオンである．認証後，Gfarm サーバは Gfarm クライアントの指示に従い，登録されている並列プログラムを実行する．並列プログラムの登録は，Gfarm クライアント，サーバにより行われ，ユーザプログラムも登録することができる．並列プログラ

ムは，世界中に存在するGfarmプールで実行される．Gfarmサーバは入力，出力のGfarmファイル名をもとにGfarmメタデータベースに問い合わせ，実行するGfarmプールのノードスケジューリングを行う．このスケジューリングは，Gfarmファイルの物理的なGfarmファイル断片の格納場所，その複製の格納場所，およびノードの状態などを元に行われる．

Gfarmクライアントは，GUI，*Gfarm*シェルと呼ばれるシェル，そしてGrid-based RPC(GridRPC)を用い，Gfarmサーバおよび Gfarmシステムにアクセスする．GridRPCは，グリッド環境における容易な遠隔手続き呼び出し方式であり，インターフェース記述言語 (IDL) の動的な読み込みによるクライアント側の簡素化と一貫性の保証，および拡張された多次元整合配列が記述可能なIDLなどを特徴としている．ユーザは，大規模ファイルのGfarmファイルシステムへの読み込み，ユーザプログラムの登録，実行，およびGfarmファイルシステムからの書き出しなどをすることができる．また，実行のモニタ，管理もすることができる．

3. Gfarm並列I/O API

Gfarm並列I/O APIはGfarmファイルシステムの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．Gfarm並列I/O APIは，それぞれのファイル断片を明示的に並列にアクセスする．並列I/O APIは，Gfarmプールのノードだけではなく，Gfarmサーバでも実行される．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，エラーのチェック，エラーメッセージのチェックがportableに簡単に可能となっている．

3.1 定　　義

Gfarmファイルと Gfarmファイル断片　*Gfarm*ファイルはGfarm URL，Gfarmファイル名で指定される論理的なファイルである．物理的には，複数のインデックス付けされた*Gfarm*ファイル断片に分割され，複数のディスクに格納される．それぞれのGfarmファイル断片はGfarm URLとインデックスで指定される．

Gfarm URLと Gfarmファイル名　*Gfarm URL*あるいは*Gfarm*ファイル名はGfarmファイルシステムにおけるGfarmファイルのパス名である．Gfarm URLは“gfarm:”で始まり，例えば以下のような形式をしている．

```
gfarm:~username/path/name
gfarm:/path/name
gfarm:path/name
```

Gfarmファイルハンドル　*Gfarm*ファイルハンドルはgfs_pio_open，gfs_pio_create，gfs_pio_open_local，gfs_pio_create_localにより割り当てられる不透明オブジェクトであり，gfs_pio_closeにより解放される．すべてのファイル操作はGfarmファイルハンドルを用いて行われる．

ローカルディスク　ローカルディスクはPCにSCSI，IDEなどで直接接続され，ネットワークディスク，リモートディスクなどに比べアクセスが速いことが期待されるが，この仮定は必要条件ではない．

3.2 エラーコード

エラーコードは整数型ではなく，エラーメッセージを含む文字列への定数ポインタである．現在は以下のエラーコードが定義されている．

```
GFARM_SUCCESS (= NULL)
GFARM_ERR_NO_MEMORY
GFARM_ERR_NO_SUCH_OBJECT
GFARM_ERR_ALREADY_EXISTS
GFARM_ERR_PERMISSION_DENIED
GFARM_ERR_INVALID_ARGUMENT
```

3.3 初期化と終了処理

```
char* gfarm_init(void);
char* gfarm_finalize(void);
```

gfarm_initはGfarmシステムの実行環境を初期化し，Gfarmメタデータベースへの接続を行う．gfarm_finalizeは実行環境を終了し，Gfarmメタデータベースへの接続を切断する．

3.4 ファイル操作

3.4.1 ファイルのオープン，作成

```
char* gfs_pio_open(char *url, int index,
    char *host, int flags, GFS_FILE *gf);
char* gfs_pio_create(char *url,
    int nfrags, int index, char *host,
    int flags, mode_t mode, GFS_FILE *gf);
```

gfs_pio_openはGfarm URL `url`とインデックス`index`で指定されたGfarmファイル断片をオープンし，新たなGfarmファイルハンドル`gf`を返す．`host`が指定されない場合は，Gfarmメタデータベースから得られる．`flags`は以下のリストのbitwise-inclusive-ORで指定する．以下の三つ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`　(hint) 必要であればファイルを複製する．

GFARM_FILE_NOT_REPLICATE (hint) 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, Gfarm ファイル断片の総数 nfrags, インデックス index で指定される新たな Gfarm ファイル断片を Gfarm プールのノード host 上にアクセスモード 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.

以下の API は, それぞれのノードが少なくとも一つの Gfarm ファイル断片を持っているという特殊ケースであるが, Gfarm においては典型的な場合の API である.

```
char* gfs_pio_set_local(
        int index, int size);
char* gfs_pio_open_local(char *url,
        int flags, GFS_FILE *gf);
char* gfs_pio_create_local(char *url,
      int flags, mode_t mode, GFS_FILE *gf);
char* gfs_pio_local_paths_get(char *url,
        int *npaths, char ***paths);
```

gfs_pio_open_local は Gfarm URL url で指定されるローカルな Gfarm ファイル断片をオープンし, 新たな Gfarm ファイルハンドル gf を返す. flags の指定は gfs_pio_open と同じである. gfs_pio_open_local により open された Gfarm ファイルは, それぞれのノードに割り当てられている一つ以上の Gfarm ファイル断片を引き続きアクセスする. gfs_pio_create_local は Gfarm URL url で指定されるローカルな Gfarm ファイル断片をアクセスモード mode で作成し, 新たな Gfarm ファイルハンドル gf を返す. gfs_pio_set_local はこれらに先立ち呼び出され, ローカルな Gfarm ファイル断片のインデックス index とファイル断片の総数 size を指定する. gfs_pio_local_paths_get は Gfarm URL url で指定されたローカルな Gfarm ファイル断片のパス名のリストと, 総数を返す.

3.4.2 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 メタデータベースのファイルサイズ, チェックサムを更新, チェックする. チェックサムはマスターファイルと複製ファイルの同一性を検証するために使われる.

3.5 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.

```
char* gfs_pio_seek(GFS_FILE gf,
        file_offset_t offset, int whence);
```

gfs_pio_seek は, ファイルハンドル gf で参照される Gfarm ファイル断片のオフセットを, whence の方向にしたがって変える.

SEEK_SET オフセットを offset バイトにする.

SEEK_CUR オフセットを現在の位置から offset バイト変える.

SEEK_END オフセットをファイルの終端から offset バイト変える.

```
char* gfs_pio_flush(GFS_FILE gf);
```

gfs_pio_flush は, ファイルハンドル gf で参照される Gfarm ファイル断片についてバッファされているデータをすべて書き出す.

```
int gfs_pio_getc(GFS_FILE gf);
int gfs_pio_ungetc(GFS_FILE gf, int c);
char* gfs_pio_putc(GFS_FILE gf, int c);
```

gfs_pio_getc は, gf から次の文字を読み, その文字を返す. ファイル断片の終端あるいはエラーの場合は EOF を返す. gfs_pio_ungetc は c を gf に戻し, 引き続きの read 操作によりその文字は読まれる. gfs_pio_putc は c を gf に書き込む.

```
char* gfs_pio_getline(GFS_FILE gf, char *s,
                      size_t size, int *eofp);
char* gfs_pio_puts(GFS_FILE gf, char *s);
char* gfs_pio_putline(GFS_FILE gf, char *s);
```

gfs_pio_getline は高々 size−1 文字を gf から読み込み, s で示されるバッファにそれらを格納する. 読み込みは EOF あるいは改行の後で終了する. 改行の場合は, '\0' が格納される. EOF だけの場合は, eofp がセットされる. gfs_pio_puts は文字列 s を gf に格納する. gfs_pio_putline は, 文字列 s と改行を gf に格納する.

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コマンドであり, Gfarm サーバおよび Gfarm プールノードで実行される.

表 1 に主要な Gfarm コマンドをあげる. この表は UNIX ファイル操作コマンドと Gfarm 管理コマンドを含んでいる. gfls, gfmkdir, gfrmdir は Garm メ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, gfchgrp,

表 1 Gfarm commands

gfls	List contents of directory
gfmkdir	Make directories
gfrm, gfrmdir	Remove directory entries
gfchmod	Change the permission mode of a file
gfchown, gfchgrp	Change file ownership
gfcp	Copy files
gfcd, gfchdir	Change working directory
gfpwd	Return working directory name
gfdf	Displays number of free disk blocks and files
gfsck	Check and repair file systems
gfimport	Import a file to Gfarm filesystem
gfexport	Export a file on Gfarm filesystem
gfdigest	Output message digest
gfrep	Replicate a Gfarm file
gfredist	Redistribute a Gfarm file on Gfarm filesystem
gfreg	Register a file to Gfarm filesystem
gfwhere	List hostnames where each Gfarm fragment is stored

gfcp 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する．gfcd, gfchdir, gfpwd は，Gfarm シェルで利用可能なコマンドである．

gfdf は，Gfarm ファイルシステムにおけるディスクの未使用ブロック，ファイル数などを表示する．gfsck は，Gfarm メタデータベースのメタデータとそれぞれの Gfarm ファイル断片の一貫性および複製ファイルとの間の一貫性チェックを検査し，可能であれば修正する．メタデータと Gfarm ファイル断片の間の一貫性は，予期しないノードやディスクの故障などにより引き起こされる．メタデータはすべての Gfarm ファイル断片が close したとき，すべてのプロセスが終了したとき，チェックポイントにおいて check-in されるため，メタデータのない Gfarm ファイル断片は消去される．

gfimport は他のファイルシステムあるいはネットワークから大規模データを import し，Gfarm ファイルシステムに分散配置する．ネットワークからのデータ転送では，GridFTP の利用を検討している．gfexport は分散されたファイルを集め，他のファイルシステムあるいはネットワークへデータを export する．gfdigest は message digest を出力する．gfrep は Gfarm ファイルの複製を作成する．gfredist は，負荷分散のために Gfarm ファイル断片の再分散を行う．gfreg は実行プログラムの登録，および Gfarm ファイルシステムへのファイルの登録を行う．ここで登録された実行プログラムは，実行時にそれぞれのローカルディスクにキャッシュされ実行される．ファイルの登録は，Gfarm 並列 I/O API を利用しないプログラムのために利用される．gfwhere は，与えられた Gfarm URL の複製を含めた Gfarm ファイル断片が格納されているホスト名を表示する．

表 2 並列 grep の予備実験結果

System	Elapsed time
UE - NetApp	100 44.6
Gfarm (UE × 2)	46.6
Gfarm (UE × 3)	30.1

(sec.)

5. 予備実験結果

Gfarm の予備評価を行うため，dual 450MHz UltraSPARC-II の Sun UltraEnterprise 220R(UE) を三台と Network-attached storage である 200GB の NetApp F720 Filer(NetApp) を Gigabit Ethernet で接続した環境で予備実験を行った．UE の OS は Solaris 7 を用いている．

表 2 は，666MB，30.9M 行の C のソースファイルを grep する時間を time(1) で計った結果である．逐次で NetApp を利用して UE で計った場合，一度目では 100 秒かかったが，二度目以降は disk cache がきいて 44 秒程度となった．Gfarm を用い，UE 二台および三台のローカルディスクを利用し，並列に grep した場合は，いずれもそれぞれ 46 秒，30 秒程度となった．Gfarm では gfimport を用いファイルを import して，gfgrep を起動している．Gfarm を利用した場合の計測には，遠隔プロセス起動などのオーバヘッドも含まれているが，実行時間の breakdown および，より大規模の評価はこれからである．しかしながら，この程度の小規模のシステムおよびデータサイズでも，ローカル I/O バンド幅および並列処理の効果が現れている．

6. 関連研究

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[1] 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[7]をサポートしており，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大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[3]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，広域，異機種では動作しない．

分散ファイルシステムは，NFS, AFS, CodaをはじめxFS, GFS[10]など様々開発されているが，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分散アクセスを可能とするものであり，キャッシュなどでアクセス効率をあげている．しかしながら，これらの方式は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．

7. まとめ

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，I/Oバンド幅を向上させるため，多数のPCクラスタのローカルディスクを利用することが有効であると考えられる．Gfarmでは，広域における数千台規模のクラスタのグループまでI/Oバンド幅をスケールさせるために，ファイル断片を分散させローカルI/Oを積極的に利用しデータ並列処理を行う．Gfarm並列I/O APIおよびGfarmコマンドにより，多数のPCのローカルディスクに単一システムイメージを与え，プログラミング，管理の複雑さを回避し，効率的な利用を目指している．これらシステムの構築には，Ninf[8),9)]などのGrid-based RPCおよびGlobus[4]などのより低レベルのグリッドサービスなどのグリッド技術，クラスタ技術が必須である．

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，詳細の性能評価および数百台規模にスケールした場合の性能評価などが必要である．また，耐故障性などを含めた完全なシステムにしていく必要もある．

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大蒔和仁部門長，ハイエンド情報技術グループのメンバ諸氏に感謝致します．

参考文献

1) Carns, P. H., III, W. B. L., 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/.

2) Collaboration, M.: Models of Networkd Analysis at Regional Centres for LHC Experiments: Phase 2 Report, Technical Report 001, CERN/LCB (2000). http://www.cern.ch/MONARC/.

3) Corbett, P. F., Feitelson, D. G., an George S. Almasi, J.-P. P., 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).

4) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *Intl J. Supercomputer Applications*, Vol. 11, No. 2, pp. 115–128 (1997).

5) Linn, J.: *Generic Security Service Application Program Interface Version 2, Update 1*, RFC-2743 (2000).

6) Matsuoka, S., Nakada, H., Sato, M. and Sekiguchi, S.: Design issues of Network Enabled Server Systems for the Grid, *Lecture Notes in Computer Science*, Vol. 1971, pp. 4–17 (2000). Proceedings of Grid Computing – GRID 2000.

7) 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.

8) Nakada, H., Sato, M. and Sekiguchi, S.: Design and Implementations of Ninf: towards a Global Computing Infrastructure, *Future Generation Computing Systems*, Vol. Metacomputing Issue (1999). http://ninf.apgrid.org/.

9) Sekiguchi, S. and Sato, M.: World-Wide Computing Infrastructure: Global and Local Partnership, *Proceedings of the second AIZU International Symposium on Parallel Algorithms / Architecture Synthesis*, pp. 25–30 (1997).

10) 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/.