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☆

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

広域で共有される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 アーキテクチャを設計し，参照実装として Gfarm を開発している．Grid Datafarm は，グリッド上の複数のクラスタを利用してペタバイト規模の広域並列ファイルシステムを構成し，その上の並列 I/O API およびファイルアフィニティスケジューリングによりスケーラブル I/O バンド幅，スケーラブル並列処理を実現する．初期性能評価では Presto III Athlon クラスタ 64 ノードを利用し，並列 I/O による書き込み，読み込みにおいて，それぞれ 1.74 GB/s，1.97 GB/s を達成した．また，並列ファイル複製では Myrinet 2000 において 23 並列で 443 MB/s 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 architecture is designed for global petascale data-intensive computing. It provides a global parallel filesystem with online petascale storage using a grid of clusters with tens of thousands of nodes. New parallel I/O APIs and file-affinity scheduling make it possible to achieve scalable I/O bandwidth and scalable parallel processing. Preliminary performance evaluation of a Gfarm reference implementation has shown scalable disk I/O and network bandwidth on the Presto III Athlon cluster. Gfarm parallel I/O write and read operations has achieved 1.74 GB/s and 1.97 GB/s, respectively, using 64 cluster nodes. Gfarm parallel file copy achieved 443 MB/s with 23 parallel streams on the Myrinet 2000.

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．1 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[9] 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カ国規模，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[17] 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 Gbps 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効率的に利用する技術，(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

†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
†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
†3 東京工業大学学術国際情報センター
Tokyo Institute of Technology
†4 株式会社 SRA
Software Research Associates, Inc.

☆ http://datafarm.apgrid.org/

ンピューティングでは，装置，計算機，研究者，可視化装置などが広域に分散するため，それらを高速接続，効率アクセスするためのグリッド技術，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[4]，PPDG[10]，iVDGL[8]，GridPP[6] 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，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（GGF）[2] で議論され，ファイル転送では，FTP にグリッド認証[5] 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[21] が提案されている．複製管理に関しては，Globus replica management[11] が主に利用されているが，いまだ研究開発段階というところである．

Grid Datafarm 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．これら典型的で大部分を占める I/O インテンシブ処理に対し，ディスク台数にスケーラブルな I/O バンド幅を提供するために，新しい広域並列ファイルシステム，新しい並列 I/O API と，分散したファイルに対するプロセスのアフィニティスケジューラを提供する．並列ファイルシステムは，ファイルをアプリケーションが操作可能なファイル断片に分割し分散配置し，ペタバイトスケールのファイルを扱う．並列 I/O API は，単一ファイルシステムイメージでそれらのファイル断片にアクセスする手段を与える．スケジューラは，典型的にはそれぞれの断片が格納されているノードにプロセスを割り当てる．これらを組み合わせて，大部分のファイルアクセスをローカル I/O アクセスとすることにより，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，グリッド認証技術，ファイル複製管理，高速並列転送などをファイルシステムレベルで統合し，グリッドサービスとして提供することによって，上記 (1)，(2)，(3) の問題の解決を計る．本論文では，アーキテクチャの提案と，その参照実装である Gfarm を利用した予備評価を行う．

2. Grid Datafarm コアアーキテクチャ

Grid Datafarm は，広域で共有される高速大規模データ処理を，複数のクラスタを利用した広域並列ファイルシステムを基本として，統一的に効率的に実現するためのアーキテクチャである．

大容量データを広域で共有し，効率的にアクセス，高速に処理するために，まず，それぞれのノードに大容量ローカルディスクを持つクラスタを高性能ディスクを持つ計算サーバとみせ，また，広域の高速ネットワークで接続された複数のクラスタ間で必要なファイルの複製を持つことにより，冗長性を高め，負荷分散を図る．それら全体を，グリッドサービスとして単一システムイメージで提供し，広域で安全かつ高効率に利用できる環境を構築する．

クラスタの各ノードに分散しているローカルディスクをまとめて，1 つの並列ファイルシステムを構築することにより，大容量ディスクを持つ並列計算サーバとすることができる．しかしながら，ペタバイト級のディスク容量とするためには，それぞれのノードに 1TB のディスクを搭載しても，数千ノード必要となってしまう．この場合，通常の並列ファイルシステムでは，複数ディスクによる I/O 性能の向上の一方で，ネットワーク性能がボトルネックとなってしまう．そこで，このボトルネックを解消するためには，ネットワークによるデータ移動を最少限とし，ノード台数にスケーラブルに増加するノード内部のローカル I/O のバンド幅を活用する必要がある．

Grid Datafarm アーキテクチャでは，I/O ノードと計算ノードを統合したクラスタを利用して，新しい並列ファイルシステムを構成する．ここでは，ファイルはアプリケーションが操作可能な任意のファイル断片に分割される．ファイル断片単位のファイルビューと，ファイル断片を格納しているノードを考慮した並列プロセススケジューラにより，アクセス局所性のあるデータ処理に関して，ノード台数にスケーラブルな I/O バンド幅と計算性能を，単一システムイメージで柔軟に可能とする．

Grid Datafarm では，広域の複数のクラスタに対して，上記システムをグリッドサービスとして構成する．広域環境における効率的なデータアクセスおよび冗長性を高めるために，それぞれの地域，クラスタ間では，必要なファイルは複製される．ファイル複製では，分割された複数のファイル断片を並列ストリームで転送することにより，ネットワークの帯域を活用する．

Grid Datafarm の参照システムである Gfarm は，

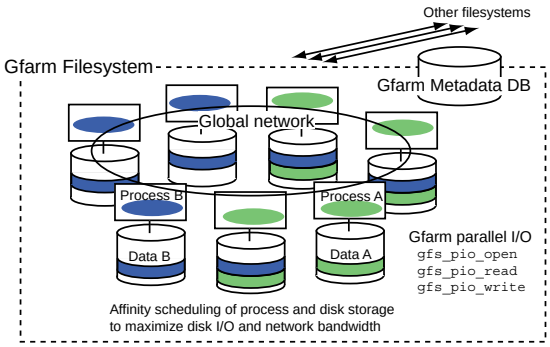

図 1 Gfarm ファイルシステム構成図
Fig. 1 Software architecture of the Gfarm filesystem.

大きく Gfarm ファイルシステム，Gfarm 並列 I/O API，Gfarm プロセスマネージャにより構成される．

2.1 Gfarm ファイルシステム

Gfarm ファイルシステムは，ペタスケール大規模データ処理のための並列ファイルシステムであり，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を利用して，グリッドサービスとして提供される（**図 1**）．Gfarm ファイルシステムは，ファイルシステムノードと呼ばれるクラスタノードのローカルディスクを利用して構成される．ファイルシステムノードは計算ノードと I/O ノードを兼ね，ノード上で I/O デーモンが動いている．また，ファイルシステムメタデータはメタサーバにより管理される．

2.1.1 Gfarm ファイル

ペタバイトスケールのファイルは，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．それらの断片はファイル断片と呼ばれる．Gfarm ファイルシステムは並列ストライピングファイルシステムを拡張したものであり，ファイル分割は，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めることができる．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 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．このとき，ファイル断片の分散，格納されているノードなどを意識することなく，Gfarm URL を用いてアクセスすることができる．

Gfarm ファイルとして，Gfarm システムにおいて実行される実行形式ファイルも格納することができる．実行プラットホームにより実行形式ファイルは異なるが，同じ Gfarm URL で，実行プラットホームに依存した実行形式ファイルを参照することができる．

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，アプリケーション側でファイル名を変更して新規ファイルとするか，read-write オープンの場合でもバージョニングを行い，内部的には別ファイルとして扱う．これは，(1) 大規模データは変更されず，read only あるいは write only のデータが多いこと，(2) 故障時あるいはデータ消失時におけるデータの復元が，一貫性維持の必要ない複製，あるいは生成ヒストリによる再計算により可能になること，などのためである．

2.1.2 ファイル複製

Gfarm ファイルはファイル断片単位の複製を持つことができる．Gfarm ファイルは write once のため，複製間において一貫性維持の必要はない．複製が複数存在する場合は，Gfarm URL により，そのうちどれかの複製がアクセスされる．

複製はノードやディスク故障時のデータ復元のためだけではなく，広域に分散して作成することにより，ファイル参照時のバンド幅，遅延の改善，および負荷分散などのために利用される．

2.1.3 Gfarm ファイルシステムメタデータ

Gfarm ファイルシステムメタデータは，ファイル状態，ファイル断片状態，ディレクトリ，複製カタログ，ファイルシステムノード状態で構成され，*gfmd* と呼ばれるメタサーバにより管理される．ファイル状態には，所有者，ファイルタイプ，アクセス許可，アクセス時刻のほか，ファイル断片の数，生成ヒストリが含まれる．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，データ生成プロセス解析などに利用される．ファイル断片状態は，ファイル ID，断片のインデックス番号，サイズ，チェックサムタイプ，チェックサムで構成される．チェックサムは複製作成時のデータ化けのチェックなどに利用される．また，複製カタログには，ファイル ID，インデックス番号，ファイルシステムノード名が含まれ，ファイルシステムノード状態には，ホスト名，アーキテクチャ，CPU 数などが含まれる．

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，メタデータの情報と実際のファイル断片の間の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．新規ファイルを作成中に，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，必

要であれば，別のノードを利用して再計算が行われる．

2.1.4 Gfarm ファイルシステムノード

それぞれのファイルシステムノードでは *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，複数ディスク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，ディスク数が多くなると，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おさえられてしまう．たとえば，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，日常的に全データを利用して処理することが必要な場合には，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，それぞれのファイルシステムノードは I/O ノードだけではなく計算ノードでもあり，複数の I/O ノードからデータをネットワークを利用して並列転送するだけではなく，それぞれのノードでプログラムを実行させることにより，ローカル I/O を積極的に利用する．ローカル I/O のバンド幅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，数万ノードで数 TB/s 程度のバンド幅を達成することができる．

2.1.5 高速ファイル転送・複製作成

Gfarm ファイルはファイル断片に分割され分散配置されるため，それぞれのファイル断片は完全に並行して複製を作成することができる．gfsd には gfsd 間において第三者ファイル転送の機能があり，それぞれのファイル断片について並列にファイルコピーすることができる．この並列ストリームによるファイルコピーにより，バイセクションバンド幅までのバンド幅を活用することができる．複製作成後はメタデータの登録を行い，複製として利用することができるようになる．

広域ネットワークでは転送遅延が大きくなるため，TCP において高いバンド幅を達成するためには，TCP のウインドウサイズ，送信，受信ソケットバッファサイズの調整，並列ストリームの利用などが必要である[22]．Gfarm では，並列 gfsd 間通信においてこれらの調整を行う予定である．また，他システムとのインタフェースとしては，GridFTP [21] の利用を検討している．

2.2 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，アプリケーションに依存した任意ブロック長のファイル分散を扱うための API である．

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（グローバルファイルビュー），特定のファイル断片（インデックスファイルビュー）など，ファイルビューによってアクセス範囲を制限することにより，アクセスの局所性を利用することができる．インデックスファイルビューでは，それぞれのファイル断片を自由にアクセスすることができるため，任意長のファイル断片を扱うことができる．

また，後述のアフィニティスケジューリングと組み合わせることにより，それぞれのプロセスが担当するファイル断片を指すローカルファイルビューを利用することができる．ローカルファイルビューは，インデックスファイルビューに対し，特定のファイル断片のインデックスを陽に指定しないものであり，ノード数，インデックス番号，ホスト名などを意識しないで，柔軟にプログラミングすることができる．ローカルファイルビューは，スケジューリングで利用したファイルのほか，新規作成ファイルに関しても適用することができる．

Gfarm では，グローバルファイルビューによる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，ローカルファイルビューを利用することにより，柔軟にローカル I/O バンド幅のスケーラビリティを利用したアクセスが可能となる．

3 章で，主要な API を紹介する．

2.3 Gfarm プロセスマネージャ

Gfarm プロセスマネージャは，スケジューリング，並列プロセス起動[24]，プロセスモニタ，プロセス異常終了検出などのプロセスの管理を行う．Gfarm では，gfmd がプロセスマネージャも兼ねている．この節ではこのうちスケジューリングについて紹介する．

2.3.1 プロセスとファイルのアフィニティスケジューリング

通常，並列プロセスの起動時には，プロセス数とホストファイルを指定してノードスケジューリングが行われるが，Gfarm ではそれ以外に，Gfarm ファイルを指定してスケジューリングすることができる．この

場合，そのファイルの総ファイル断片数と同数の並列プロセスが，基本的にはファイル断片が格納されているノードにスケジューリングされる．

ファイル断片は複製を持つことができるため，スケジュールされるノードには任意性があり，また，ノードの負荷状況，ディスクの利用状況などによっては，ファイル断片を持たないノードにもスケジュールすることができるため，様々なスケジューリング手法が考えられる．

図 1 では，プロセス A はデータ A が分散しているノードに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 の分散しているノードにスケジューリングされるが，データ B は 3 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，複製を含めて 6 ノードに分散している．よって，プロセス B は，ノードの負荷状況などにより，そのうち 3 ノードにスケジューリングされている．このように，複製は耐故障性のためではなく，負荷分散にも利用される．

2.4 障害時などにおけるファイル復元と再計算

大規模データ処理においては，部品点数が増加するため，障害によりファイルを消失してしまう可能性が高くなる．そこで，失われたファイルを復元するための手段が重要となる．

Gfarm ファイルシステムでは，ファイルは複製を持つことができ，ファイルが消失してしまった場合も，ファイルの複製が存在する場合は，Gfarm URL によりその複製が参照されるため，問題なくアクセスすることができる．

複製がない場合は，再計算により復元することになる．Gfarm では，ファイル生成時に利用されたプログラム名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．プログラムおよび実行時に参照されるファイルが，すべて Gfarm ファイルシステムに存在する場合，ファイル生成時に利用したのと同じプログラム，同じ引数を利用し再計算することができ，ファイルを復元することができる．

GriPhyN の Virtual Data Concept[4] は，アプリケーション依存の問合せに対し，存在するデータを読み込むだけではなく，存在しないデータも動的に計算するという概念である．Gfarm は，この概念を動的再計算によるファイル復元を利用して，ファイルシステムレベルでサポートすることができる．つまり，アプリケーション依存の問合せから Gfarm ファイル名への写像を定義し，ファイル生成に必要なプログラムを登録し，生成ヒストリをあらかじめ適切に定義することにより，この概念を実現することができる．

ただし，タイミングに関するバグが存在するなど，実行されるプログラムに非決定性がある場合には，再計算しても同じファイルが生成されるとは限らない．そのため，1 度目の計算時に計算したチェックサムを利用して同じファイルかどうか確認する．また，再計算による一貫性を保証するために，あるプロセスがファイルを write あるいは read-write でオープンしている間は，別のプロセスはそのファイルのオープンに失敗する．

Gfarm ファイルは，仮にすべての複製が消去されてしまっても，そのメタデータは消去されないため，消去された Gfarm ファイルに生成ヒストリが存在し，実行に必要なすべてのファイルを参照あるいは再生成することができれば，再計算により復元することができる．

上記により，CPU，ディスク，ネットワークなどに障害が起きた場合は，メタデータが存在する限り，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．また，メタデータに関しては，基本的に複製を分散して持つことにより，耐故障性と広域におけるスケーラビリティを実現する．ただし，メタデータは write once ではなく更新が起こるため，それらの複製間の一貫性制御が必要となる．

2.5 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）[5] 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，それぞれの資源は単一認証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，gfmd，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大きくなってしまう．そのため，Gfarm では複数の認証方式を併用し，安全なクラスタ内では期限付き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．

2.6 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．そ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．データインテンシブ処理では，この grep 処理に相当する処理を，ペタバイトスケールのデータに対して実行したいという要

求がある[15]．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 処理を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ローカルファイルビューで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ローカル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．このとき，入力ファイル，出力ファイルのそれぞれのファイル断片の大きさは任意であり，入力ファイルのファイル断片が格納されるノードで自動的に実行されることに注意されたい．また，ファイルの分割方式は，この例からも分かるように基本的にアプリケーション依存とすることができる．

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，最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．イベントごとに処理したデータは入力データと同じローカルファイルビューで書き出すことにより，スケーラブル処理となる．処理後のデータは，処理の正しさを知るため，あるいは解析結果表示のために，ヒストグラムなどの作成が必要となるが，これらの処理は，グローバルファイルビューによるアクセスか，MPI を利用した 1 対 1 通信，集合通信により実現される．

3. Gfarm 並列 I/O API

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，文献 3)，19)，25) を参照されたいが，そのうち主要なものは以下である．

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 1 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，必ずそのとお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ち，それぞれ独立したファイル位置を示すことができる．

3.1.2 ファイルビュー

ファイルビューは，並列プロセスがファイルを参照するときにアクセスすることのできるファイル位置の集合である．

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，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ローカル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．ローカルファイルビューでは，ファイル断片のインデックスは明示的に指定されないが，プロセスランクと同じインデックスとなる．

以下の API は，ファイル断片のインデックスを明示的に指定する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．すでにある 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，それらの値はファイルシステムメタデータ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 ファイルシステムメタデータ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を利用しているが，独立した数千，数万のデータベース，ファイルに対して，まとめて同様の操作をしたいというような場合に，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とができる．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 1 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，ローカル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 の場合はローカルファイルビューとする．そのときのもとのコード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．

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，文献 3)，19)，25) 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 ファイルシステム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，ファイルシステムメタデータと 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．

gfreg は Gfarm ファイルシステムへ実行形式ファイルを含むファイルの登録を行う．gfrep は Gfarm

ファイルのファイル断片単位の複製を指定されたホストリストに並列に作成する．

5. 実装の状況

現在，すべての Gfarm 並列 I/O API と，必要最小限の Gfarm コマンド[19] が実装され，Linux では最小限のシステムコールトラップが実装されている．それぞれのファイルシステムノードでは I/O デーモンである gfsd が動き，遠隔ファイル操作，遠隔プロセス起動，gfsd 間の第三者ファイル転送，ロードモニタが実装されている．メタサーバとしては OpenLDAP サーバを利用している．ユーザ認証は，鍵の配送が信頼された環境における期限付き共有秘密鍵方式が実装されている．プロトタイプソフトウェアは東工大 Presto III および産総研のクラスタに配備されている．

今後，動的に複製を作成することによる負荷分散と耐故障性の向上，生成ヒストリを利用した再計算によるファイル復元を実装する予定である．また，グリッド環境における GSI による認証，高速プロセス起動[24]，耐故障性と広域における性能向上のための複数メタサーバの連携も実装する予定である．

6. 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．ノードの CPU は dual AMD Athlon MP 1.2 GHz であり，100 Mbps 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 メタサーバは LDAP サーバを用い，Fast Ethernet で接続される dual Pentium III 500 MHz のノードを利用した．それぞれの OS は Linux 2.4 である．

Gfarm が対象としているアプリケーションでは，ほとんどのアクセスが大規模ファイルに対するローカルファイルビューのアクセスとなり，その基本性能評価として，以下では並列 I/O 性能，および並列ファイル複製のバンド幅の評価を行った．

6.1 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分散させて測定した．このとき，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，まず 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断片を並列に作成する．次に，gfs_pio_write により

```
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
Fig. 2 An excerpt to measure Gfarm parallel I/O bandwidth.

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 ファイルシステムメタデータは更新される．

読み込みに関しては，まず起動時に，並列書き込みで作成した Gfarm ファイルを利用してプロセススケジューリングを行う．次に，gfs_pio_open によりメタデータを参照し，その Gfarm ファイルをオープンして，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のローカルファイルビューとする．一般的には，それぞれのプロセスがみているファイル断片は，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ローカルディスクへの複製作成となる．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明示的にスケジューリングした．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 64K バッファ領域に対して繰り返し書き込み，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 64 ノードを利用し，640 GB データを読み書きしたときの 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 768 MB を大きく超える 10 GB 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もとにしており，メタデータのアクセス，チェックサム計算のオーバヘッドなどが含まれている．

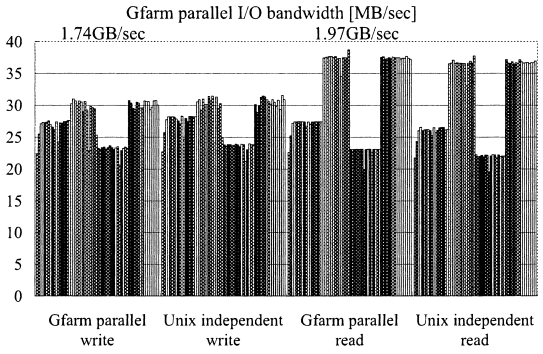

図 3 Gfarm 並列 I/O 性能
Fig. 3 Gfarm parallel I/O performance.

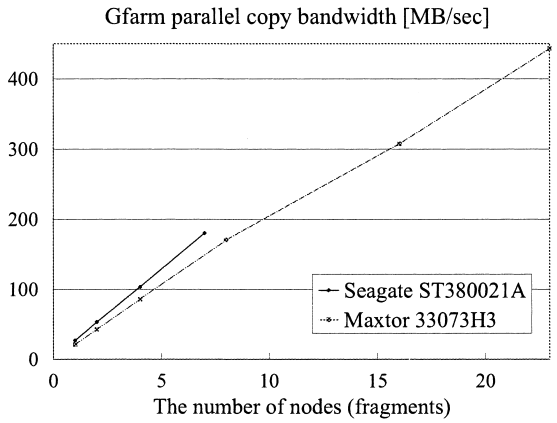

図 4 Gfarm 並列ファイル複製性能
Fig. 4 Gfarm parallel file replication performance.

Gfarm 並列書き込みでは 640 GB のデータに対し，合わせて 1.74 GB/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 27.2 MB/s である．Presto III クラスタのノードのディスクは Seagate ST380021A あるいは Maxtor 33073H3 が利用され，図 3 におけるそれぞれのノードのディスク性能の差となっている．

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 1.77 GB/s となり，Gfarm 並列書き込みにおけるメタデータのアクセス，チェックサム計算のオーバヘッドはほぼ無視できることが分かった．

Gfarm 並列読み込みでは総計で 1.97 GB/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 30.8 MB/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平均で 29.9 MB/s であり，Gfarm 並列読み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当ての問題と考えられ，性能差はほぼ無視できる．

これらの結果より，64 ノードまでは並列 I/O 性能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．

6.2 並列ファイル複製

Gfarm ファイルはファイル断片に分割され，Gfarm ファイルシステムノード上のディスクに分散配置されている．ファイル断片はそれぞれ並列に転送することができ，Gfarm ファイルは並列ストリームで複製される．

図 4 に，それぞれのファイル断片のサイズが 10 GB の Gfarm ファイルを，Gfarm コマンド `gfrep` を利用して複製したときのバンド幅を示す．ファイルサイズはファイル断片の数，つまり分散されるノード数に応じて変わる．

`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する．ファイル複製は，それぞれの gfsd 間の並列ストリームにより行われる．

並列ファイル複製にあたり，ネットワークは Myrinet 2000 を利用した．Myrinet 2000 は 130 MB/s 程度のバンド幅があるが，それぞれのファイル断片の転送はディスク I/O バンド幅に制限され，ストリームあたり，Seagate で 26 MB/s，Maxtor で 21 MB/s であった．図 4 より，23 並列ストリームで 443 MB/s を達成していることが分かる．

図 4 における `gfrep` の性能は `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すべてのオーバヘッドを含んでいるが，複製のバンド幅は，少なくとも Presto III において，23 並列ストリームまではスケールしている．

7. 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[16] 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，ファイル断片に対するファイルビュー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により，ファイルのネットワーク転送量をおさえ，ローカル I/O によるスケーラビリティを利用することができる．

Linux クラスタのローカルディスクを利用したクラスタファイルシステムとして PVFS[13] 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 MPI-IO もサポートされており，さらに mount して UNIX/POSIX I/O API を利用することもできる．一方，Gfarm は，ストライピングを自由なファイル断片の分散としたところ，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，複製カタログ，再計算の生成ヒストリをファイルシステムメタデータとして扱い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なっている．

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[7] は分散階層型ストレージシステムであり，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[12]，GFS[18] 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，複数からの小規模ファイル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および大規模ファイルの読み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[21]，Legion I/O[23]，Kangaroo[20] など様々なシステムが提案されている．GridFTP は FTP に GSI[5]，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蔽すると同時に，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蔽している．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超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．Globus replica management は，Gfarm の一部分の機能を提供するもので，複製の作成と，論理ファイル名による複製の問合せを行うことができる．Gfarm のメタデータサーバの一部に相当するものは Replica Catalog とよばれ，現在は，上記の論理ファイル名から複数の複製への物理的な位置を持っているだけであるが，今後の GGF における議論で，Gfarm のファイルシステムメタデータで持っているような情報も含まれることも十分に考えられ，その場合はメタサーバ部分は統合されるかもしれない．しかしながら，Gfarm はメタデータを利用して，大規模ファイルに対するインデックスファイルビューおよびファイルアフィニティスケジューリングを提供し，広域で共有されるペタバイトスケールのオンラインディスク，高帯域，高性能，高信頼のデータ処理を提供する．

8. 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，および広域における効率的な共有の実現が重要となる．Grid Datafarm では数万台規模の PC クラスタのローカル I/O を利用することにより，ペタバイトスケールを超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ディスクにより並列ファイルシステムを構築し，それらに分散したファイルに対するデータ並列処理のための Gfarm 並列 I/O API と，分散ファイル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，数万台規模の複数クラスタに対

して，Gfarm ファイルシステムを構築するためには，効率的な共有およびノード，ディスク，ネットワークなどの故障に対応する必要がある．Grid Datafarm では，それらに対応するため，ファイルは複製を持つことができ，それら複製はメタデータで一貫して管理される．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．失われたファイルに対しては，メータデータに含まれる生成ヒストリを利用して再実行し，復元することができる．

予備性能評価では，Presto III Athlon クラスタ 64 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 1.74 GB/s，並列読み込みでは 1.97 GB/s を達成し，また並列ファイル複製では Myrinet 2000 において 23 ストリームを用い 443 MB/s を達成した．ローカル I/O を利用したローカルファイルビューにおける並列書き込み，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はほぼ隠れ，スケーラブルな性能を示した．

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，さらにノードを増やした場合の性能評価，広域における複数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，現在 Grid Datafarm のシミュレータを開発中であり，負荷分散および耐故障性を増す効率的な複製作成アルゴリズム，スケジューリングアルゴリズムの評価などを行っていく予定である．

Grid Datafarm は，アーキテクチャ的には数 TB/s 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，ご討論を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いたします．

参 考 文 献

1) EU DataGrid Project. http://www.eu-datagrid.org/
2) Global Grid Forum. http://www.gridforum.org/
3) Grid Datafarm. http://datafarm.apgrid.org/
4) Grid Physics Network. http://www.griphyn.org/
5) The Grid Security Infrastructure Working Group. http://www.gridforum.org/security/gsi/index.html
6) The GridPP Project. http://www.gridpp.ac.uk/
7) HPSS: High Performance Storage System. http://www.sdsc.edu/hpss/
8) International Virtual Data Grid Laboratory. http://www.ivdgl.org/
9) The Large Hadron Collider. http://www.cern.ch/lhc/
10) Particle Physics Data Grid. http://www.ppdg.net/
11) Allcock, B., Bester, J., Bresnahan, J., et al.: Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing, *Proc. IEEE Mass Storage Conference* (2001).
12) Anderson, T.E., Dahlin, M.D., Neefe, J.M., et al.: Serverless network file systems, *Proc. 15th ACM Symposium of Operating Systems Principles*, pp.109–126 (1995).
13) Carns, P.H., Ligon III, W.B., Ross, R.B., et al.: PVFS: A Parallel File System for Linux Clusters, *Proc. 4th Annual Linux Showcase and Conference*, pp.317–327 (2000).
14) Corbett, P.F., Feitelson, D.G., Prost, J.-P., et al.: Parallel file systems for the IBM SP computers, *IBM Systems Journal*, Vol.34, No.2, pp.222–248 (1995).
15) Gray, J.: The World Wide Telescope: Mining the Sky (2001). Plenary talk of SC2001.
16) 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
17) MONARC Collaboration: Models of Networked Analysis at Regional Centres for LHC Experiments: Phase 2 Report, Technical Report CERN/LCB-001, CERN (2000). http://www.cern.ch/MONARC/
18) Soltis, S.R., Ruwart, T.M. and O'Keefe, M.T.: The Global File System, *Proc. 5th NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies* (1996).

http://www.globalfilesystem.org/.
19) Tatebe, O., Morita, Y., Matsuoka, S., et al.: 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
20) Thain, D., Basney, J., Son, S.-C., et al.: The Kangaroo Approach to Data Movement on the Grid, *Proc. 10th IEEE International Symposium on High Performance Distributed Computing*, pp.325–333 (2001).
21) The Globus Project: *GridFTP: Universal Data Transfer for the Grid*. http://www.globus.org/datagrid/deliverables/C2WPdraft3.pdf
22) Tierney, B.L.: *TCP Tuning Guide for Distributed Application on Wide Area Networks*. http://www-didc.lbl.gov/tcp-wan.html
23) White, B.S., Grimshaw, A.S. and Nguyen-Tuong, A.: Grid-Based File Access: The Legion I/O Model, *Proc. 9th IEEE International Symposium on High Performance Distributed Computing*, pp.165–173 (2000).
24) 岩崎 聖, 松岡 聡, 曽田哲之, 平野基孝, 建部修見, 関口智嗣：Grid 環境における大規模クラスタ向けジョブマネージメントアーキテクチャの実装及び性能評価, 情報処理学会研究報告 2001-HPC-89, pp.37–42 (2002).
25) 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).

(平成 14 年 1 月 29 日受付)
(平成 14 年 5 月 22 日採録)

建部 修見（正会員）

昭和 44 年生．平成 4 年東京大学理学部情報科学科卒業．平成 9 年同大学大学院理学系研究科情報科学専攻博士課程修了．同年電子技術総合研究所入所．理学博士．独立行政法人産業技術総合研究所グリッド研究センター．グリッドコンピューティング，並列数値アルゴリズム，並列計算機システムの研究に従事．日本応用数理学会，ACM 各会員．

森田 洋平

昭和 35 年生．昭和 58 年筑波大学第一学群自然学類卒業．昭和 63 年同大学大学院物理学研究科物理学専攻博士課程単位取得退学．筑波大学準研究員，日本学術振興会特別研究員等を経て，平成 3 年高エネルギー物理学研究所入所．理学博士．文部科学省高エネルギ加速器研究機構計算科学センター．素粒子実験，大容量データ解析システムの研究に従事．日本物理学会，IEEE 各会員．

松岡 聡（正会員）

昭和 38 年生．昭和 61 年東京大学理学部情報科学科卒業．平成元年同大学大学院博士課程から，同大学情報科学科助手，同大学情報工学専攻講師を経て，平成 8 年東京工業大学情報理工学研究科数理・計算科学専攻助教授．平成 13 年 4 月東京工業大学学術国際情報センター教授，平成 14 年国立情報学研究所客員教授併任．理学博士．高性能システム，並列処理，グリッド計算，クラスタ計算機，高性能・並列オブジェクト指向言語処理系等の研究に従事．平成 8 年度情報処理学会論文賞，平成 11 年情報処理学会坂井記念賞受賞．ソフトウェア科学会，ACM，IEEE 各会員．

関口 智嗣（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 59 年筑波大学大学院理工学研究科修了．同年電子技術総合研究所入所．独立行政法人産業技術総合研究所グリッド研究センター．以来，データ駆動型スーパコンピュータ SIGMA-1 の開発等の研究に従事．並列数値アルゴリズム，計算機性能評価技術，ネットワークコンピューティングに興味を持つ．市村賞受賞．日本応用数理学会，SIAM，IEEE 各会員．

曽田 哲之

昭和 39 年生．平成 2 年名古屋大学理学部物理学科卒業．同年株式会社 SRA 入社，現在に至る．並列計算機システムに関する技術開発に従事．