2004－HPC－99　(25)
2004／7／31

広域仮想ファイルシステム Gfarm v2 の設計と実装

建部修見[†]　曽田哲之[††]　関口智嗣[†]

複数組織からなるグリッド環境において，高信頼のファイル共有，高性能分散並列データ・コンピューティングを実現するため，グリッド・データファーム・アーキテクチャを提案している．Gfarm v2 は，そのアーキテクチャの実装であり，分散並列処理の特徴をいかしたまま，POSIX 標準 API をサポートする広域仮想ファイルシステムを提供する．本論文では，Gfarm v2 の設計と実装について述べる．

Gfarm v2: Design and Implementation of Global Virtual File System

Osamu Tatebe,[†] Noriyuki Soda[††] and Satoshi Sekiguchi[†]

Grid Datafarm architecture is designed for facilitating reliable file sharing and high-performance distributed and parallel data computing in a Grid across administrative domains by providing a global virtual file system. Gfarm v2 is an attempt to implement a global virtual file system that supports complete set of standard POSIX APIs, having the parallel and distributed data computing feature of Grid Datafarm architecture. This paper discusses the design and implementation of Gfarm v2 that provides secure, robust, scalable and high-performance global virtual file system.

1. はじめに

グリッド技術の研究開発，標準化が進むにつれ，組織をまたがる CPU やディスクなどの資源の共有が可能となってきた．資源の共有にあたり，個人認証，権限の設定，情報漏洩の防止などセキュリティの問題は徐々に解決しつつあり，資源を共有するための土台ができようとしている．今後は，それら広域に分散する資源をいかに効率よく利用するかが問題となる．

我々のグループでは，KEK，東工大，東大などとの共同研究により，広域に分散する資源を利用して，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のためのグリッド・データファーム・アーキテクチャを提案した[13]．本アーキテクチャの大きな特徴は，(広域に分散する) PC クラスタのクラスタノードなどの CPU とディスクの両方を利用して，広域仮想ファイルシステムと並列分散処理を提供するところにある．

高エネルギー物理学，天文学などのデータ・インテンシブ・コンピューティングでは，数多くのファイルに対し同じプログラムで処理したい，という要求がある．例えば，次々と生成される実験データ，観測データに対し，同様の処理を行う，あるいは数多くの乱数の種からシミュレーションデータを生成する，などである．これらの処理は，PC クラスタと高性能共有ファイルサーバを別々に準備する従来型のアプローチでは，PC クラスタとファイルサーバ間のネットワーク性能がボトルネックとなってしまい，大規模なシステムとすることが困難であった．

それに対し，上記提案では，PC やクラスタノードのローカルディスクの作業領域（ローカルファイルシステム）を統合し，仮想的なパス名で POSIX インターフェースによりファイル参照を可能とする．この時，利用者やアプリケーションはどの作業領域にファイルが格納されているかを意識する必要がない．さらに，ファイルの集合を一つのファイル（Gfarm ファイル，スーパファイル）として扱うことができ，そのファイル集合の分散配置された構成ファイルを，分散配置されているノードで分散並列処理することを可能とする（ファイル・アフィニティ・スケジューリング，ローカル・ファイル・ビュー）．

また，それぞれの構成ファイルは自由にファイル複製を作成し，分散配置させることができる．これにより，耐故障性の実現と，ファイルアクセス，分散並列処理に関する負荷分散を実現する．

我々のグループは，上記アーキテクチャを実現するため，プロトタイプ実装（以後 Gfarm v1）を行い，性

† 産業技術総合研究所グリッド研究センター
Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology (AIST)
E-mail: {o.tatebe,s.sekiguchi}@aist.go.jp
†† (株) SRA
Software Research Associates, Inc.
E-mail: soda@sra.co.jp

能評価，アプリケーションへの適用をすすめてきた．Gfarm v1 では，大規模データ・インテンシブ・コンピューティングのため，一度作成されたら変更されないという大規模ファイルの特徴などを利用して，ファイル入出力の API としては POSIX 標準で提供される API のサブセットを提供していた．これにより，高性能なデータ処理という要求は満たされるが，その一方で汎用性を失っており，広範囲のアプリケーションに適用を進める上でしばしば問題となっていた．

そこで，本論文では，グリッド・データファームの特徴をいかしたまま，POSIX 標準 API をサポートする広域仮想ファイルシステムの実現を目指した Gfarm v2 における設計と実装について議論する．広域仮想ファイルシステムは，グリッド・ファイルシステムとも呼ばれ，広域に分散するファイルシステム，ストレージシステムを統合したファイルシステムである．ファイルは実際にはどこかのファイルシステム，ストレージシステムに格納されるが，その格納位置，ファイル名とは無関係に，仮想的な階層ディレクトリによりアクセスされるため，仮想ファイルシステムと呼ばれる．

Gfarm v2 における広域仮想ファイルシステム（以後 Gfarm ファイルシステム）は，グリッド環境全体（仮想組織）におけるネットワーク・ファイルシステムである．複数組織間において，既存アプリケーションによりファイル共有が安全に高信頼に可能となるほか，ファイルシステム上のデータを高速にアクセス，高速処理することが可能となる．

2. 関 連 研 究

高性能のファイルシステムに関する研究開発は，おもに PVFS[8]，Luster[4]，Google File System[9] などのクラスタ・ファイルシステムと，CXFS[1]，Storage Tank[6]，Red Hat Global File System[5] などの SAN ファイルシステムがある．いずれも，複数のストレージをネットワークで接続して，ファイルシステムを提供するという点では共通しているが，ストレージの利用方式が異る．クラスタ・ファイルシステムでは，それぞれのストレージはそれぞれのファイルシステムで管理されるが，SAN ファイルシステムでは，それぞれのストレージはブロックレベルでアクセスされる☆．

ここで挙げたクラスタ・ファイルシステム，SAN ファイルシステムは，いずれもファイルはアプリケーションで利用するデータ長とは無関係な固定長のブロックで分散される．それにより，単一ファイルの連続読出し時に，複数ディスクからの読出しとなり，性能の向上が期待される．また，ファイルアクセスの均一化，分散化による性能の向上もある．しかしながら，その一方で，データ・インテンシブ・コンピューティングにおける「数多くのファイルに対し同じプログラムで処理する」場合には，ファイルアクセスの局所性が利用できず，効率的な処理とならない．

それに対し，Gfarm ファイルシステムでは，固定ブロックサイズではなく，ファイル単位で分散格納する．これにより，単一ファイルに対するアクセス性能は向上しないが，複数のファイルを一斉に処理するような場合，ファイルアクセスの局所性を利用した，スケーラブルな分散並列データ処理が可能となる．また，ファイル複製を作成することにより，集中アクセス時の性能低下も防ぐことができる．

3. Gfarm v2 の設計

Gfarm v2 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のためのグリッド・データファーム・アーキテクチャ[13] による分散並列処理を可能としつつ，完全な広域仮想ファイルシステムを提供し，

- より広範囲のアプリケーションへの適用，
- 日常的な利用

を可能とすることを目標とする．ここで提供する広域仮想ファイルシステムは，NFS[7),12] や Andrew File System (AFS) などのネットワークファイルシステムの代替となりうる，POSIX 標準 API をサポートする頑強で信頼性のあるファイルシステムである．プロトタイプ実装である Gfarm v1 に対しては，主に以下の点に関して改善，機能向上を計る．

ファイルシステム機能　ファイルの読み書きオープン，ファイルロックの対応

頑強性　メタデータと実際に格納されるファイルの間の一貫性保持

セキュリティ　メタデータのセキュリティとユーザ，グループの管理

分散並列処理　ファイルモデルの一般化

3.1 ファイルの読み書きオープン，ファイルロックの対応

Gfarm v1 では，一度作成されたら変更されないという大規模ファイルの特徴を利用して，ファイルの複製を作成し，分散配置していた．CPU 負荷，応答時間などにより適宜適切なファイル複製を選択することにより，ファイルのアクセス集中を防ぎ，またノードやネットワーク障害に備えることができる．そのため，基本的には，ファイルの書き込みオープンは，`creat` と同等の処理，つまり新たに作成される場合，`O_TRUNC` により長さ 0 に切り詰められる場合しか実装されていなかった．

しかしながら，この制限は Gfarm ファイルシステムを利用する上でしばしば大きな制限となっており，例えば Gaussian のチェックポイントファイルの生成

☆ SAN ファイルシステムでは iSCSI や network block device なども利用され，必ずしも SAN (Storage Area Network) が利用されるとは限らない．

をはじめ，ファイルの更新を行うアプリケーションからの利用を困難なものとしていた．Gfarm v2 では，ファイルの複製がある場合でも，ファイルの書き換え，追加など読み書きオープンをサポートし，この制限をなくする．またファイルロックもサポートする．

3.2 メタデータと実際に格納されるファイルの間の一貫性保持

Gfarm v1 では，アプリケーションから呼び出される Gfarm I/O ライブラリがメタデータの操作と，ファイルアクセスを行っていた．このため，アプリケーションがファイル書込み途中で強制終了，異常終了した場合，メタデータを更新することができず，メタデータの内容と実際に格納されているファイルの間で一貫性が失われてしまっていた．Gfarm v2 ではこの問題に対し，一貫性が問題となるメタデータ操作を I/O ライブラリではなく，後述するファイルシステムサーバにより行うことで，一貫性の保持を実現する．

また，実際に格納されているファイルが，仮想ファイルシステムを利用したアクセス以外の方法で，変更，移動，消去などされた場合，仮想ファイルシステムの管理している情報との不整合が生じてしまう．

Gfarm v1 では，ファイルは利用者が操作可能なファイルとして格納されていたため，直接，変更，消去することが可能であった．これに対し，Gfarm v2 では，全てのファイルを，ファイルシステムサーバの所有とし，他ユーザへの参照，書込み許可を出さないことにより，実データを保護し，不当な更新を防ぐ．

3.3 メタデータのセキュリティとユーザ，グループの管理

ファイルシステムではディレクトリ，ファイル単位に所有者，グループを設定することができ，読込み，書込み，実行などのアクセス制御を設定することができる．そのアクセス制御を一般化したアクセス制御リスト (ACL) による制御もある．さらに，それらの制御はファイルやディレクトリの所有者などにより変更可能である．

ここでのアクセス制御は，実際のファイルなどに対するアクセス制御だけではなく，仮想ファイルシステムの階層ディレクトリ構造に対するアクセスにも適用される必要がある．例えば，`/foo/bar/baz` というパス名に対し，もしも `/foo/bar` のディレクトリに実行権限がなければ，`baz` の情報は提供してはならないし，読込み権限がなければ，ディレクトリの内容を提供してはならない．

Gfarm v1 では，仮想ファイルシステム・メタデータの管理のために LDAP サーバを直接利用していた．そのため，上記のような細かいアクセス制御はできていなかった．これに対し，Gfarm v2 では，これらの制御を可能とする．

3.4 ファイルモデルの一般化

グリッド・データファームでは，並列ファイルアクセス，並列データ処理のために，ファイル断片の集合からなる Gfarm ファイルを扱うことができる．Gfarm ファイルは Gfarm 並列 I/O API の `gfs_pio_open` によりオープンされ，その後，ファイルアクセスのためのファイル・ビューを変更することにより，それらのファイル断片に自由にアクセスすることができる[13]．

Gfarm v1 では上記を実現するため，仮想ファイルシステムで Gfarm ファイルと仮想ディレクトリを管理していた．基本的なファイルの単位は Gfarm ファイルであり，Gfarm ファイルは複数のファイル断片を管理し，それぞれのファイル断片はその複製が格納されている場所を管理する．そのため，

- ある特定のファイル断片を複数の Gfarm ファイルに含める，
- Gfarm ファイルのファイル断片として別の Gfarm ファイルをもつ

ことなどができなかった．このため，例えば 14) における観測データの一次処理に関して，フラット画像作成では 10 並列であるが，作成したフラット画像による一次処理では n 並列，だが，処理するデータは同じ $10 \times n$ のデータというような場合，Gfarm v1 ではそれぞれ別々の Gfarm ファイルを準備して，複製の管理対象とならないデータのコピーを二つもつことになる．さらに，$10 \times n$ 並列の処理を行う場合は，n 並列のための Gfarm ファイルを対象に 10 回並列プロセスを起動するか，$10 \times n$ 並列のための Gfarm ファイルをさらに準備する必要があり，この場合，また複製の管理対象とならないデータのコピーが必要となる．

これに対し，Gfarm ファイルを特別なファイルではなく，仮想ディレクトリで表すことにより，柔軟なファイル集合の管理が可能となる．このとき，ファイル断片は，その仮想ディレクトリに含まれるファイルとなる．

共有されるファイル断片に関しては，仮想ファイルシステム上のシンボリックリンク，あるいはハードリンクにより同一のファイルを共有することができる．なお，ファイル断片はファイル断片単位で複製が作成され，管理されている．

また，さらに，Gfarm ファイルの指定方法として，単一の仮想ディレクトリ名（あるいは単一ファイル名）による指定だけではなく，ワイルドカード，正規表現などによる指定方法の拡張を行う．この時，ワイルドカードや正規表現にマッチする複数のファイル，複数の仮想ディレクトリからなる Gfarm ファイルが指定されることとなる．

これにより，それぞれに仮想ディレクトリを作成するのではなく，on-the-fly でファイルの集合を Gfarm ファイルとして指定し，アクセスすることが可能となり，利便性が向上する．

4. Gfarm v2 の実装

Gfarm v2 における Gfarm ファイルシステムは，Gfarm v1 と同様に，仮想ファイルシステム・メタデータサーバ（以降メタデータサーバ）と，ファイルシステムを提供するファイルシステムノードで構成される．ファイルシステムノードでは，遠隔ファイルアクセスを実現するため，ファイルシステムサーバが動作する．

アプリケーションやクライアント・プログラムは，ファイルシステムノードあるいはクライアント PC などで実行され，Gfarm I/O ライブラリを通して Gfarm ファイルシステムに対するアクセスを行う．

以下では，それぞれに新たに必要となる実装を議論する．

4.1 メタデータサーバ

メタデータサーバは，仮想ファイルシステムを構成するために必要なメタデータを管理する．

4.1.1 仮想ファイルシステム・メタデータ

仮想ファイルシステムのメタデータのうち主要なものとして，仮想ファイルシステムの階層的なディレクトリ構成，仮想ディレクトリとファイルの情報，実際に格納されているファイルの位置情報がある．ファイルは複製され，異る場所に格納されるため，ファイルの位置情報はその複製の数だけ存在し，複製カタログとも呼ばれる．

Gfarm v1では，複数のファイル断片からなる Gfarm ファイルの管理のため，Gfarm ファイルの情報とファイル断片の情報をもっていた．Gfarm v2 では，ファイルモデルの一般化のため，仮想ファイルシステム的には Gfarm ファイルの情報を特別に管理する必要がなくなった．

また，Gfarm v2 では，読み書きファイル・オープン，ファイルロックをサポートするため，メタデータとしてオープン，ロックの状態を保持する．後述するが，これらの状態の変更は，ファイルシステムサーバにより一貫して行われる．そのため，アプリケーションのクラッシュやプログラムエラーではロックが保持されたままとなるなどの一貫性の崩れはない．

4.1.2 仮想ファイルシステムレベルのユーザ，グループ管理

組織をまたがるグリッドの環境においては，組織や個々の資源の独立性が重視され，権限の設定はそれぞれの組織，資源でなされる．グリッド環境全体に対しては，基本的に個人を表す名前しかないため，それぞれの組織で個人を認証し，権限を設定することになる．

個人認証ではなく，グループとして認証，権限設定を行う方式として，CAS (Community Authorization Service)[10] が提案されている．この場合，それぞれのサイトは，個人を登録する代りに全面的に CAS を信頼することになる．CAS により認証されたかどうか，CAS によりどういう権限を与えたられたかに従い，利用者に権限を与える．この場合，先のそれぞれの組織の独立性は失われるが，管理面，利用面での煩雑さを解消することができる．

CAS をファイルシステムのアクセス制御に利用することは不可能ではないが，この場合，仮想ファイルシステムが提供する，ディレクトリベースのアクセス制御を CAS で行うということになる．つまり，CAS に対し仮想ファイルシステムのメタデータサーバの機能を持たせる必要がある．

将来的には，この方向性での設計，実装を検討する必要があるが，CAS は現在標準化のプロセスにのっていないこと，現状の CAS の実装は未成熟なことにより，Gfarm v2 では，メタデータでグループの管理を行うこととする．これにより，独自管理となるが，仮想ファイルシステムレベルのユーザ，グループ単位のアクセス制御，ACL の実現を図る．

4.1.3 メタデータのセキュリティ

ファイルシステムのディレクトリ情報は，たとえそのファイルの内容が分からなくとも，誰がどういうファイルを所持しているか等を表しており，機密性の高い情報である．Gfarm v2 では，ディレクトリ単位の細かなアクセス制御を実現するため，既存のデータベースサーバを直接利用するのではなく，個人認証，細かなアクセス制御を行うためのメタデータサーバの開発を行い，そのメタデータサーバのバックエンドとして既存のデータベースを利用する．

4.1.4 ファイルシステムノード情報とノードの選択

Gfarm v1 では，ファイルシステムノード名の管理をメタデータで行い，ノードの選択では，Garm I/O ライブラリが必要なときに必要なノードに対し CPU 負荷などの情報を UDP で問い合わせていた．しかしながら，この方法では，ノードの負荷情報が共有されないため，無駄が多い．また，それぞれが得た情報により個別に割り当てを行うため，同時刻の同じ情報を元にすると割り当てがぶつかってしまう．

そのため，Gfarm v2 では，メタサーバでファイルシステムノードの負荷情報を集中管理し，情報の共有を行うとともに，ノードの選択を行い，同時刻の同じ情報により集中して割り当てられてしまう現象の回避を図る．

4.2 ファイルシステムサーバ

ファイルシステムサーバは，遠隔のファイルシステムに対するアクセスを実現するために，ファイルシステムノードで動作するサーバである．

Gfarm v2 における大きな変更点は，ファイルシステムサーバによる，実データの更新に一貫したメタデータの更新と，実データの保護である．これにより，Gfarm ファイルシステムの頑強性が向上する．

4.2.1 メタデータの一貫した更新

新規ファイルを作成する場合，ファイルのオープン，

ファイル・ビューの設定，データの読み書き，ファイルのクローズが行われる．ファイルのオープンでは，メタデータをファイル・オープン状態とする．ファイル・ビューが設定され，実際のファイルの読み書きの前には，設定されたファイル・ビューとファイルシステムノードの CPU 負荷に応じて，実際にファイルが格納されるファイルシステムノードが決定される．そのファイルシステムノードに対しファイルの読み書きを行った後，ファイルのクローズ時には，作成されたファイルの情報をメタデータに反映させる．

これら一連の操作において，途中で操作が中断してしまうと，メタデータに不正なデータが残ったり，メタデータの内容と異る作成中のファイルが残ったりしてしまい，メタデータとの一貫性が失われてしまう．Gfarm v1 では，この操作をアプリケーションがリンクしている Gfarm I/O ライブラリにより行っていたため，ファイル書込み途中でアプリケーションが強制終了，異常終了した場合，必ず一貫性が失われてしまっていた．そのため，Gfarm v1 では，ファイル参照時やファイルシステム・チェック時などに，メタデータと実データの一貫性のチェックを行い，ごみの消去，不正なメタデータの修正を行っていた．

この問題に対し，Gfarm v2 では，これらのファイル操作とメタデータの更新をファイルシステムサーバにより一貫して行う．ファイル書込み途中で，アプリケーションが終了した場合，ファイルシステムサーバがその終了を検知し，書込み途中の状態をメタデータに反映させる．

4.2.2 実データの保護

Gfarm v2 では，全てのファイルをファイルシステムサーバの所有とし，他ユーザへの参照，書込み許可を出さないことにより，実際に格納されているファイルに対する不当なアクセスを防ぐ．この場合，基本的には全てのアクセスはファイルシステムサーバ経由でソケットにより行われることになる．そのため，ファイルシステムノードで動作しているアプリケーションから，自ノードのファイルシステムを直接参照する場合の性能が低下する可能性がある．そのため，自ノードのファイル参照に関しては，ファイルシステムサーバ経由でファイル・ディスクリプタを転送（ディスクリプタ・パッシング）し，性能低下を防ぐ．

また，ユーザプログラムの異常終了などによるコア・ファイルは，ユーザ権限で生成される．そのため，各ユーザ権限で書込み可能なディレクトリを別途準備し，各プロセスはそのディレクトリをカレント・ワーキング・ディレクトリとして実行し，コア・ファイルの生成を可能とする．

4.3 Gfarm I/O ライブラリ

Gfarm v2 では，Gfarm I/O ライブラリで，新たに読み書きモードのファイル・オープンとファイルロックのサポートを行い，読み書きに関して制限のない Gfarm ファイルシステムへのアクセスを実現する．

4.3.1 読み書きモードのファイルオープン

Gfarm ファイルシステムでは，ファイルは自由に複製が作成され，複製は自由に配置される．Gfarm v1 では，ファイルの修正，更新はサポートしなかったため，ファイル複製間で一貫性の崩れはなかったが，読み書きモードのサポートにより，一貫性の保持が必須となる．

Gfarm v2 における，ユーザーに対する一貫性の保証は下記の通りとなる．

(1) ファイルロックを行わない場合，AFS と同様に，書込みプロセスがファイルをクローズした後にオープンしたプロセスにのみ，そのファイルの内容の反映が保証される．
(2) ファイルロックを行っている場合，ファイルロックしているプロセス間では，ファイルロックした範囲について，最新の内容がアクセスされることを保証する．ただしファイルロックは advisory ロックのため，ファイルロックしていないプロセスが書込みを行うなどの場合には，この保証はない．

上記の一貫性の保証を行うため，ファイル・オープン時におけるファイルの複製選択は以下のように行う．

(1) 読み込みオープン要求
  (a) 任意のファイル複製を選択する．
(2) 書込みオープン要求
  (a) そのファイルを書込みオープンしているプロセスが存在する場合，書き込みオープンしているプロセスが参照しているのと同じファイル複製を選択する．
  (b) そのファイルを読込みオープンしているプロセスが存在する場合，読み込みオープンしているプロセスのうち任意の一つが参照しているファイル複製を選択する．
  (c) そのファイルをオープンしているプロセスが存在しない場合，任意のファイル複製を選択する．

上記の選択は，実際にはメタデータサーバにより行われる．このようにファイル複製を選択することにより，異るファイル複製について同時に書込みオープンされることはない．

メタデータサーバは，書込みオープンが生じた場合，ファイルへの書込み後，クローズされた時点で，書込み先として選択されたファイル複製以外のファイル複製を消去する．オープン時ではなくクローズ時に消去するのは，書込みオープンしたが，実際には書き込まれなかったという場合があるからである．複製の消去に関しては，既に読み込みオープンされているファイル複製も対象となるが，そのファイルが既にオープンされていればクローズされるまではアクセスできるため問題はない．

また，複数のプロセスが異るファイル複製に対して読込みオープンした後に，書込みオープンされた場合，あるいは，書込みオープンされてもクローズされるまでの間に読込みオープンされた場合は，書込みオープンされたものとは異るファイル複製へアクセスしつづける場合がある．書込みオープンしてクローズする前に，書込み内容を他プロセスに反映させるためには，ファイルロックを利用する必要がある．

4.3.2 ファイルロック

Gfarm v2 では，POSIX における advisory ロックをサポートする．ファイルロックでは，ファイルの一部あるいは全体に対して，読込みロック，書込みロックのサポートを行う．

基本的な実装方針としては，ファイルロック時には，全プロセスが同じファイル複製を参照するようにする．また，ファイル内容の一貫性保持のため，ロックされている範囲に関しては，クライアント・ライブラリによるキャッシュを行わない．

読込みのファイルロック要求時に，他プロセスが同ファイルを書込みオープン，あるいは書込みロックを保持し，別ファイル複製を参照していた場合，その別ファイル複製を参照するように変更し，読込みロックの要求を出す．

5. 国際標準にむけて

グリッド技術では，広く利用を進めるために，標準化がかかせない．Gfarm v1 および Gfarm v2 の経験を活かし，GGF[2] において，広域仮想ファイルシステムの標準化をすすめるために，グリッド・ファイルシステム WG を提案を行い，2004 年 1 月に WG が承認された[3]．WG の目標は，Open Grid Services Architecture (OGSA) に基づく仮想ファイルシステム・ディレクトリ・サービス (VFDS) の標準仕様書の策定と，OGSA による VFDS やその他のサービスを利用したグリッド・ファイルシステムのアーキテクチャに関する仕様書の策定である．

VFDS の仕様書では，本論文におけるメタデータサーバのインターフェースを定めるものである．referral による VFDS 間の連携も考えられ，DNS のようなディレクトリの分散管理が行われる．憲章のスケジュールでは，2005 年 3 月には VFDS の仕様書の完成を予定している．なお，VFDS に関しては NFSv4[11] に対するグローバル名前空間の提供という連携も議論されている．

6. まとめと今後の課題

Gfarm v2 における広域仮想ファイルシステムの設計と実装ついて議論した．Gfarm v2 では，グリッド・データファームの分散並列処理の特徴を残したまま，NFS を代替可能，かつ複数組織間で安全で高信頼にファイル共有が可能となる Gfarm ファイルシステムの実現を目標としている．今後，実装を進め，性能評価，ソフトウェアの成熟化を行っていく予定である．

また，今回，設計を議論した以外にも，メタデータ・アクセスの性能向上，ノード割り当ての性能向上，およびデータの保全に関しても設計を行っている．今後，データ保全の保証，自動複製作成などの研究開発を行い，信頼性の向上を目指す．さらに，国際標準として進めている OGSA ベースの研究開発も同時に進めていく予定である．

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) *CXFS*. http://www.sgi.com/products/storage/cxfs.html.
2) *Global Grid Forum*. http://www.ggf.org/.
3) *Grid File System WG*. https://forge.gridforum.org/projects/gfs-wg/.
4) *Lustre*. http://www.lustre.org/.
5) *Red Hat Global File System*. http://www.redhat.com/software/rha/gfs/.
6) *Storage Tank*. http://www.almaden.ibm.com/StorageSystems/file_systems/storage_tank/index.shtml.
7) Callaghan, B., Pawlowski, B. and Staubach, P.: *NFS Version 3 Protocol Specification* (1995). RFC 1813.
8) Carns, P.H., Ligon III, W.B., Ross, R.B. and Thakur, R.: PVFS: A Parallel File System for Linux Clusters, *Proc. of the 4th Annual Linux Showcase and Conference*, pp. 317–327 (2000).
9) Ghemawat, S., Gobioff, H. and Leung, S.-T.: The Google File System, *Proc. 19th ACM Symposium on Operating Systems Principles (SOSP-19)* (2003).
10) Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S.: A Community Authorization Service for Group Collaboration, *Proc. of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks* (2002).
11) 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.
12) Sun Microsystems: *NFS: Network File System Protocol specification* (1989). RFC 1094.
13) 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).
14) 山本 直孝, 建部 修見, 関口 智嗣: グリッドデータファームにおける天文学データ解析環境の構築と評価, 先進的計算基盤システムシンポジウム SACSIS2004 論文集, pp. 233–240 (2004).