GSI-SFS：グリッドのためのシングルサインオン機能を有するセキュアファイルシステム

武田伸悟[†] 伊達進[†] 下條真司[††]

グリッドでは科学者や研究者が仮想組織を形成することで国境や組織の枠を越えてコンピュータのリソースを共有することができる．このような共同研究の形態は飛躍的な科学技術の進展をもたらすと期待されている．科学技術計算を行うアプリケーションの多くはデータの入出力にファイルを利用するため，科学者や研究者はグリッド環境において利便性の高いファイルアクセス手段を要求している．また，機密性の高いデータを扱う医療などの研究分野では強固なセキュリティが必要とされている．本論文では利便性とセキュリティを両立する分散ファイルシステム GSI-SFS を提案する．GSI-SFS は既存の分散ファイルシステム SFS をグリッドのセキュリティ基盤である GSI で拡張したものである．GSI-SFS ではシングルサインオンで，グリッド上のファイルをローカルファイルと同様に扱うことが可能である．また，ネットワークを流れるすべてのデータは SFS の機能により自動的に暗号化，検証され機密性と整合性が保護される．GSI-SFS により，グリッドに関する深い知識を持たない科学者や研究者が安全かつ容易にデータを共有することが可能となり，機密データを扱う分野にもグリッド導入が進むことが期待される．

GSI-SFS: A Secure Filesystem with Single Sign-on Functionality for the Grid

Shingo Takeda,[†] Susumu Date[†] and Shinji Shimojo[††]

Grid technology allows scientists and researchers to share their computer resources across organizational or national boundaries by forming virtual organizations. These large-scale collaborations are expected to make rapid advances in science and technology. Since most of current sci-tech applications use files for data I/O, scientists and researchers demand a user-friendly file access method. Robust security is also demanded in some fields, such as medicine where patient privacy is handled. In this paper, we describe GSI-SFS, a secure distributed filesystem with a single sign-on functionality. GSI-SFS is developed by extending SFS, an existing distributed filesystem, with GSI, a security infrastructure for the grid. GSI-SFS allows users to access files on the grid as if they were local. Furthermore, all data transferred over network are automatically encrypted and verified by SFS. GSI-SFS facilitates scientists and researches sharing data securely, and promotes introduction of the grids to the organizations handling confidential data.

1. 背景と目的

近年のネットワークの普及と高速化にともない，広域分散したコンピュータ上のリソースを共有可能とするグリッド（Grid）[1] への注目が高まっている．グリッドでは科学者らが国境や組織の枠を越えて仮想組織（Virtual Organization: VO）を自由に形成し，データやプロセッサなどのリソースを共有することができる．このような大規模な共同研究の形態は，科学技術の飛躍的な進展をもたらすものと期待されている．

グリッド技術は Global Grid Forum（GGF）[☆] において現在標準化が進められている．GGF には研究機関からだけでなく多数の IT 企業からの研究者も参加しており，グリッド技術のビジネス応用への期待も大きい．

Globus Toolkit [2]（以下，Globus と略記する）は Globus Alliance [☆☆] によってオープンソースで開発されているグリッド構築のためのミドルウェアである．

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†† 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University

☆ http://www.ggf.org/
☆☆ http://www.globus.org/

Globus は GGF の標準に従った多くの機能を実装しており，デファクトスタンダードとして数多くの組織で導入され実験がなされている．Globus はミドルウェアであり，単体で利用するのではなく，提供されている Application Programming Interface（API）を利用してユーザがアプリケーションを開発することを前提としている．Globus の提供する API を使用してアプリケーションの拡張や新規開発を行うためにはグリッドに関する専門知識が必要であり，これらの作業が専門知識を持たない科学者にとって大きな負担となっている．

グリッドで共有される重要なリソースの 1 つにデータがある．現在の科学技術計算アプリケーションではデータの入出力にファイルが最も頻繁に使用されている．科学者らは Globus の提供する API を使用してグリッド上のファイルにアクセスするのではなく，それらがあたかも手元のコンピュータにあるかのように利用したいと考えている．そこで，このようなアクセスを実現する利便性の高いファイルアクセス手段が強く求められている．

一方で，ファイルアクセスにおいては強固なセキュリティも必要とされることが多い．テーラーメイド医療やゲノム創薬といった個人データを扱う研究分野においてはデータの機密保護が必要不可欠である．また観測データや計算結果だけでなく，実行ファイルや設定ファイルなど改竄リスクの高いファイルも扱われるため，整合性を保護することも必要である．

グリッドにおけるファイルアクセス手段にはこれらの利便性とセキュリティの両方の要件を満たすものが必要である．しかし，一般にセキュリティを高めることはユーザにより多くの制限を課すことにつながり，利便性とセキュリティの両立は難しい．本研究はセキュリティ強度の低下を最小限とし，かつユーザ利便性の高いファイルアクセス手法を提案することでグリッド技術，さらには科学技術の発展に貢献することを目的とする．

本論文では，まず 2 章で既存のデータおよびファイルアクセス技術の利便性とセキュリティについて調査する．3 章では本論文で提案するファイルシステムの設計と実装について述べ，4 章でそれをセキュリティ，利便性，スループットの面から評価する．5 章では提案手法の応用事例を紹介し，6 章で今後の方針について述べ，7 章で本論文をまとめる．

2. データおよびファイルアクセス技術

グリッドでの大規模なデータ共有環境の実現に関しては，今日までに多くの研究がなされている．また，従来から比較的小規模なデータ共有環境の構築には分散ファイルシステムが利用されてきた．本章ではグリッド技術と分散ファイルシステムの，ファイルアクセスの利便性とセキュリティについて述べる．**表 1** は本章における比較をまとめたものであり，以下参照されたい．

2.1 グリッドにおけるファイルアクセス

GridFTP[3),4)] は，従来の FTP をデータ転送効率とセキュリティの向上に焦点をおいて拡張したプロトコルである．広域ネットワークでは伝送遅延が大きいため GridFTP では遅延による転送効率の低下を防ぐために，複数 TCP ストリームを使用した並列転送，データチャネルの再利用などの機能が追加されている．GridFTP は RFC 2228 “FTP Security Extensions” [5)] を拡張するプロトコルで，データを暗号化，検証して転送する機能を持つと規定されている．ユーザとホストの認証には Grid Security Infrastructure（GSI）[6)] が使用される．

GSI は Globus で実装されている，グリッドのためのセキュリティ機構である．GSI では公開鍵基盤（Public Key Infrastructure: PKI）に基づいた認証を行うため，Globus で構築されたグリッドの各ユーザとホストはそれぞれ認証局（Certificate Authority: CA）か

表 1 既存手法の利便性とセキュリティ
Table 1 Convenience and security of existing methods.

	ファイルアクセス		データグリッドミドルウェア		分散ファイルシステム		
	GridFTP (Globus)	Parrot (GridFTP)	SRB	Gfarm	NFS	AFS, Coda	SFS
利便性							
POSIX ファイル I/O	×	△デバッガ	△動的ライブラリ	△静的ライブラリ	○	○	○
必要環境	–	Linux	Linux/Solaris	UNIX	OS に依存	OS に依存	NFS
認証	GSI	GSI	GSI など	GSI など	ホスト間	パスワード	公開鍵
管理権限が不要	○	○	○	○	×	×	○
セキュリティ							
通信の暗号化と検証	×	×	○	○	×	○	○

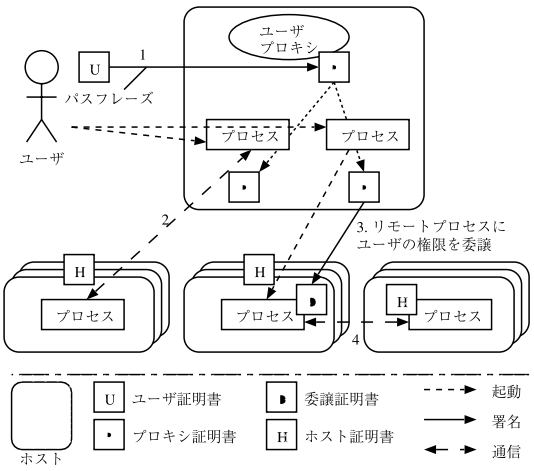

図 1 GSI のシングルサインオン機能
Fig. 1 Single sign-on functionality of the GSI.

ら発行された電子証明書を持つ．GSI では電子署名を応用してシングルサインオン（Single sign-on）を実現している．GSI のシングルサインオンの大まかな仕組みを**図 1** に示す．ユーザはグリッドへのログイン時にパスフレーズを入力してプロキシ証明書に署名し，それをユーザプロキシに登録する（図 1–1）．アプリケーションのプロセスはこのユーザプロキシからプロキシ証明書を取得して他のホスト上のプロセスと認証を行うため，異なるアプリケーションを使用するたびにユーザがパスフレーズを入力する必要がない（図 1–2）．さらに GSI では他のホストで実行したプロセスに委譲証明書を発行することでユーザの権限を委譲することができ（図 1–3），第三者ホスト上の認証おいてもシングルサインオンが有効である（図 1–4）．

Globus はファイル転送サービスとして GridFTP の主な機能を実装している．GridFTP は広域ネットワークでの転送効率が高いため，Globus で実装されている GridFTP はデータのレプリケーション[7] など，多数のデータ管理サービスで採用されている．Globus では GridFTP を使用するためのコマンドラインツールと API を提供しているが，コマンドラインツールでは部分的なファイルアクセスが行えないなど非効率である．したがって効率的なアクセスのためにはアプリケーションから API を呼び出すことが求められる．本論文執筆時点では，Globus の GridFTP は RFC 2228 を実装しておらず，データの暗号化と検証は行わない．

Parrot[8] はネットワーク上のファイルにローカルファイルと同様に，すなわち Portable Operating System Interface for UNIX（POSIX）のファイル入出力 API でアクセスすることを可能にするミドルウェアである．Parrot は HTTP，FTP，GridFTP，Network Storage Technology（NeST），Castor Remote File I/O（RFIO）など様々なプロトコルに対応している．`parrot` コマンドを使用してアプリケーションを起動すると，Parrot は Linux の ptrace デバッギングインタフェースを通じてアプリケーションのシステムコールをトラップし，上述したプロトコルとの相互変換を行う．そのため，Parrot は Linux でのみ利用できる．Parrot の GridFTP は Globus の実装を使用しており，GSI の認証で利用できるが通信の暗号化と検証は行わない．

2.2 データグリッド構築のためのミドルウェア

Storage Resource Broker（SRB）[9] はファイルやデータベースなど異種のデータを統合的に管理し，共有できるデータグリッド構築のためのミドルウェアである．メタデータカタログやレプリケーションの機能も備えており，機能が豊富であることから BIRN☆や NASA☆☆をはじめとする多くのプロジェクトで採用されている．SRB 上のファイルへの POSIX API でのアクセスを可能とするために，SRB は動的リンクライブラリを提供している．ユーザが `setprel` コマンドを実行すると，以後このライブラリがアプリケーションのファイル入出力関数呼び出しをトラップし，SRB へのアクセスに変換する．この機能はライブラリを静的リンクしているアプリケーションでは利用できず，また Linux の一部のディストリビューションと Solaris に限って利用できる．SRB は GSI の認証で利用でき，GSI を使用して通信の暗号化と検証を行える．

Gfarm は Grid Datafarm[10] の参照実装であり，データインテンシブコンピューティングに特化したミドルウェアである．Grid Datafarm は粒子加速器から出力される膨大な観測データなど，ペタバイトスケールのデータを高速に並列処理することを目的としたアーキテクチャである．Gfarm 上のファイルへの，POSIX API でのアクセスを可能とするために，Gfarm はアプリケーションのシステムコールをトラップする静的リンクライブラリを提供している．これを利用するためにはアプリケーションを再ビルドして静的リンクライブラリをリンクする必要がある．Gfarm も SRB と同様に GSI の認証で利用でき，GSI を使用して通信の暗号化と検証を行える．

2.3 分散ファイルシステム

ネットワーク上のファイルへのネットワーク透過的

☆ http://www.nbirn.net/Resources/Users/Applications/SRB/index.htm
☆☆ http://www.ipg.nasa.gov/ipgusers/power/srb.html

なアクセスを実現するものに分散ファイルシステムがある．分散ファイルシステムはオペレーティングシステム（Operating System: OS）に依存した機能を使用することが多いため，多種の OS で動作する可搬性の高い分散ファイルシステムを開発することは難しい．その反面，Parrot，SRB，Gfarm のようなシステムコールをトラップする方式に比べて，分散ファイルシステムは適用可能なアプリケーションが多い．

現在，最も多くの OS が実装している分散ファイルシステムは NFSv3（以下，NFS）である．NFS の認証はホストとホストの間で行われ，管理特権を持つユーザのみがマウント操作を実行できる．NFS では通信の暗号化と検証は行わない．

Andrew File System（AFS）[11] と，AFS から派生しモバイル向けの改良が行われた Coda [12] は通信の暗号化と検証を行う代表的な分散ファイルシステムである．AFS と Coda ではパスワードによるユーザごとの認証を行うが，クライアントが接続するサーバを決定できるのはクライアントの管理特権を持つユーザ（root ユーザ）のみである．NFS ではマウント先のディレクトリは決められていないのに対して，AFS では `/afs`，Coda では `/coda` にすべてのリモートディレクトリがマウントされ，これらの下のディレクトリ階層は管理ドメイン内で同じである．ユーザは管理ドメイン内のどのホストにログインしても同じディレクトリ構造が利用でき，シングルファイルシステムイメージが実現されている．

Self-certifying File System（SFS）[13] はセキュリティ，可搬性，柔軟性を重視して NFS 上に開発された分散ファイルシステムである．SFS は NFS の RPC を暗号化，検証して中継することで NFS のセキュリティを改善している．SFS は TCP で NFS を中継する仕組みであり，NFS に対応した Linux，FreeBSD，OpenBSD，Solaris，OSF/1 など多くの UNIX 系 OS で動作が確認されており可搬性が高い．また，SFS ではユーザごとの公開鍵認証が行われ，非特権ユーザ（root 以外のユーザ）が任意のサーバに接続できる点で NFS とは異なる．SFS では以下に示す形式の self-certifying pathname と呼ばれるパスを使用する．

`/sfs/@`*Hostname*`,`*HostID*`/`*Path*

Hostname は SFS サーバのホスト名，*HostID* はサーバの公開鍵などのハッシュ値，*Path* はサーバ上のパスである．*HostID* はサーバのなりすましを防ぐために付加されている．この self-certifying pathname はインターネット上のファイルを一意に指し示す．このパスを導入することで，SFS は中央サーバを置かない非集中型であるにもかかわらずグローバルなシングルファイルシステムイメージ（以下，グローバルファイルシステムイメージ）を実現している．ユーザが self-certifying pathname にアクセスするとオンデマンドでマウントが行われるため，あらかじめマウント操作を行う必要はない．SFS では複数ユーザでクライアントを共有した場合においてもそれぞれのユーザが排他的な `/sfs` を持ち，他のユーザがどのサーバにアクセスしているかさえも知ることができず機密性が高い．

3. 設計と実装

Globus や SRB，Gfarm などが提供するミドルウェア独自の API を使用してアプリケーションの拡張や新規開発を行うことは，特にグリッドに関する専門知識を持たないユーザにとって大きな負担となる．POSIX API でのアクセスに対するユーザの需要は大きく，Parrot，SRB，Gfarm がそれぞれ異なる方式でこれを実現している．しかし，これらの方式では適用可能なアプリケーションが限られている．Parrot は `parrot` コマンドの子プロセス，SRB はライブラリが動的リンクされているアプリケーション，Gfarm は再ビルドによって静的リンクライブラリをリンクしたアプリケーションでのみ利用できる．それに対して，分散ファイルシステムは OS に組み込まれて動作するのですべてのアプリケーションに適用される．しかし，2.3 節であげた分散ファイルシステムはグリッドのような大規模な共有環境を想定して設計されておらず，そのままグリッドに導入することはできない．そこで，本論文ではユーザの負担を軽減し利便性を改善することを目的とした，グリッドで利用できる分散ファイルシステムを提案する．

3.1 機能要件

1 章で述べたように，科学者は利便性とセキュリティ要件の両方の充足を要求する．そのようなユーザの要求を分析し，新しいファイルシステムには以下の機能が必要であると考えた．

シングルサインオン グリッドではユーザは意識することなく多数のホストを使用する．異なるホストに接続するたびにパスフレーズの入力を求めることはできず，ユーザが一度パスフレーズを入力した後は認証を意識することなく利用できるシングルサインオンの機能が不可欠である．

グローバルファイルシステムイメージ グリッドにおける分散計算では，しばしばプログラムを実行するホストが動的に割り当てられる．プログラムがグリッド上のどのホストで実行されるかによってディレクト

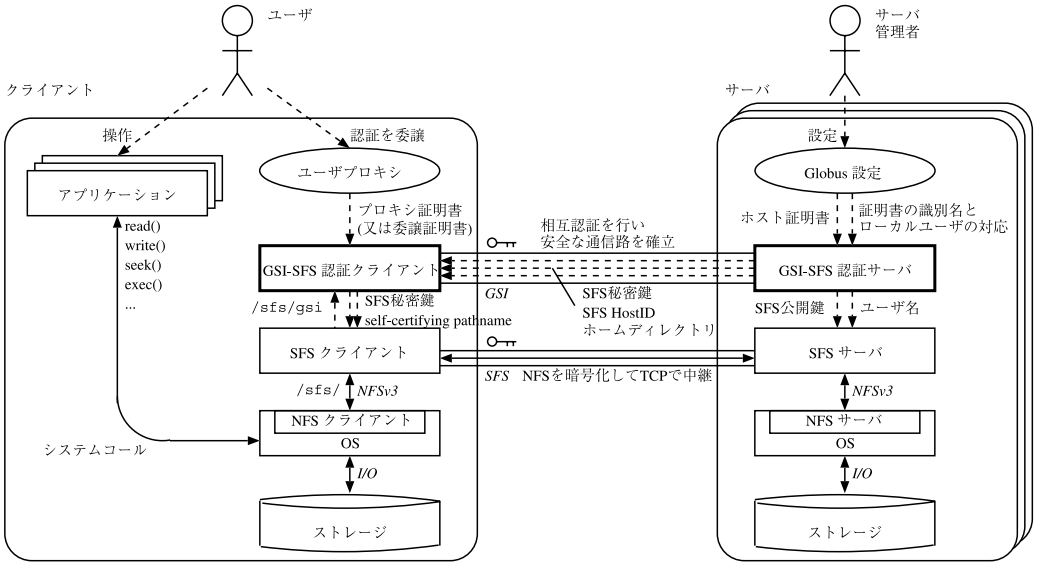

図 2 GSI-SFS のアーキテクチャ
Fig. 2 Architecture of the GSI-SFS.

リ構造が異なってはならない．ホストとユーザは複数の VO に属することがあるため，VO に依存しないグローバルな名前空間が望ましい．

非特権ユーザによるオンデマンドマウント 利用する可能性のあるすべてのサーバをあらかじめマウントしておくことは非効率的かつ非現実的である．ユーザやアプリケーションのアクセスに応じてオンデマンドで認証，マウントを行う必要がある．グリッドのユーザにホストの管理特権を与えることは運用上困難であるため，非特権ユーザでも任意のサーバに接続できる必要がある．

機密性と整合性の保護 グリッドは医療など機密性の高いデータを扱う分野でも導入が検討されており，データの機密保護が必要である．また，グリッドにおける分散計算では実行ファイルや設定ファイルなど，改竄リスクが高いファイルも転送されるため整合性の保護も必要である．

高い可搬性 グリッドは異質なリソースで構成される環境であり，多種の OS が混在する．そのため可能な限り OS に依存しないように実装すべきである．

2 章の調査結果より，これらの機能のうちシングルサインオンは GSI で，残りの機能は SFS で実現されているといえる．そのためファイルシステムを一から開発するのではなく，SFS をベースとし，GSI の拡張を行うことですべての機能を実現することを考えた．

3.2 アーキテクチャ

本節では本論文で提案する GSI-SFS のアーキテクチャについて述べる．GSI-SFS は SFS を拡張し，GSI のシングルサインオンで SFS を利用できるようにした分散ファイルシステムである．GSI-SFS のアーキテクチャを**図 2** に示す．拡張した部分は SFS からは独立した GSI-SFS 認証サーバ（以下，認証サーバ）と GSI-SFS 認証クライアント（以下，認証クライアント）で，これらは GSI の認証で SFS を利用することを可能にする．現時点では SFS への変更は行っていない．認証サーバと同クライアントは Globus 2.2 の GSI を使用して開発した．Globus 2.0 や本論文執筆時点で最新の Globus 3.0.2 に付属の Globus 2.4.3 でも動作を確認している．

SFS には，クライアントが動的に self-certifying pathname を取得するための仕組みが備わっている．SFS クライアントは，`/sfs/` で始まるが `/sfs/@` で始まらないパスにアクセスすると，クライアントに登録されているプログラム（`certprog` と呼ばれる）を起動して `/sfs/@` で始まる self-certifying pathname を取得する．現在の GSI-SFS の実装では，この `certprog` の機能を利用することで SFS を変更することなく GSI の認証を実現している．GSI-SFS では以下の形式のパスを使用する．

`/sfs/gsi/`*Hostname*`/`*Path*

GSI-SFS においては GSI の相互認証でサーバの成りすましを防ぐことができるため，2.3 節で示した SFS の HostID を使用せずにグローバルファイルシステムイメージを実現している．インストール時に作成され

るシンボリックリンクを通して以下の形式のパスで，より直感的にアクセスすることも可能である．

`/gsisfs/`*Hostname*`/`*Path*

ホームディレクトリは一般にユーザが自由に読み書きできることから多用されるが，グリッドは異質な環境でありホームディレクトリの位置がホストによって異なる．GSI-SFS では以下の形式のパス，

`/sfs/gsi-home/`*Hostname*`/`*Path_from_homedir*

またはシンボリックリンクを通してより直感的なパス，

`/gsisfs-home/`*Hostname*`/`*Path_from_homedir*

を使用して統一的にホームディレクトリを利用できる．

GSI-SFS での認証の流れを以下に示す．

(1) アプリケーションが `/sfs/gsi` で始まるパスにアクセスすると，SFS クライアントは `certprog` として登録されている認証クライアントを起動する．このとき *Hostname* と *Path* が認証クライアントにコマンドライン引数として与えられる．

(2) 認証クライアントは与えられた *Hostname* で表される認証サーバに TCP で接続する．

(3) 認証クライアントはユーザプロキシからプロキシ証明書（または委譲証明書）を取得し，認証サーバはサーバ上に保存されているホスト証明書を取得する．そして，これらクライアントとサーバは GSI の機能を使用し証明書による相互認証を行って安全な通信路を確立する．

(4) 認証サーバは `grid-mapfile` を参照してユーザ証明書の識別名（Distinguished Name: DN）とローカルユーザを対応づける．ユーザ証明書の識別名はプロキシ証明書（または委譲証明書）の識別名から知ることができる．`grid-mapfile` はユーザ証明書の識別名とローカルユーザの対応が記述された Globus で共通の設定ファイルである．

(5) 認証サーバは対応づけられたローカルユーザのための SFS 鍵ペアを生成し，公開鍵を SFS サーバへ登録する．鍵の生成処理には推測されにくい乱数を生成する必要があるため数秒程度の時間を要する．そのため認証サーバは一定期間（現在の実装では 24 時間）生成した鍵ペアをキャッシュしておき，同じユーザからの要求が再度あればキャッシュしている鍵ペアを使用し，鍵の生成は行わない．

(6) 認証サーバは生成した秘密鍵，SFS サーバの HostID，サーバ上のユーザのホームディレクトリのパスの 3 つの情報を認証クライアントに送信する．これらの情報は GSI の機能により暗号化，検証され安全に転送される．

(7) 認証クライアントは SFS クライアントに受信した秘密鍵を登録し，受信した SFS サーバの HostID と引数として与えられた *Path* から self-certifying pathname を作成，出力する．このとき，`/sfs/gsi/` でなく `/sfs/gsi-home/` で始まるパスにアクセスしていた場合は受信したホームディレクトリのパスも付加して出力する．

(8) SFS クライアントは認証クライアントから出力された self-certifying pathname と，登録されている秘密鍵を使用して SFS サーバに接続する．

以上より，GSI の認証によるシングルサインオンで SFS が利用可能となる．これら一連の処理は自動的に行われ，ユーザが意識する必要はなく，SFS と同様にオンデマンドなマウントが実現されている．

4. 評　　価

本章ではセキュリティと利便性の面から GSI-SFS を評価する．また，SFS のスループットの評価も行い，GSI-SFS がどのような科学技術計算に適しているかについて述べる．

4.1 セキュリティ

GSI-SFS ではネットワークを流れる SFS の秘密鍵，SFS サーバの HostID，ホームディレクトリのパスは GSI の機能によって暗号化，検証される．そして，NFS の通信は SFS によって暗号化，検証され中継される．このため GSI-SFS ではネットワークを流れるすべてのデータが暗号化，検証され，盗聴や改竄からデータが保護される．GSI-SFS でも SFS と同様に各ユーザは排他的なグローバルファイルシステムイメージを持つ．そのため，クライアントを複数のユーザで共有した場合においてもユーザは他のユーザのファイルシステムイメージを知ることはできない（**図 3**）．グリッドでは高性能なホストを多数の組織で共有することがあるが，そのような場合においても GSI-SFS の安全性は高い．

GSI-SFS でも他の多くの GSI を使用するアプリケーションと同様に `grid-mapfile` に従ってグリッドのユーザをローカルユーザに対応づける．GSI-SFS 自体にはアクセス制御の機能はなく，管理者は OS が提供するファイルのパーミッションと NFS のエクスポート設定でアクセス制御を行う．そのため，GSI-SFS はアクセス制御の柔軟性に欠ける．また，GSI ではすべてのホストの `grid-mapfile` を設定する必要があり，

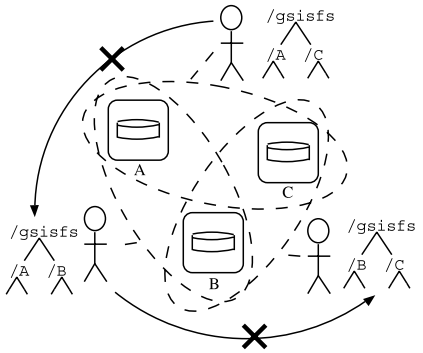

図 3 排他的なグローバルファイルシステムイメージ
Fig. 3 Exclusive global filesystem images.

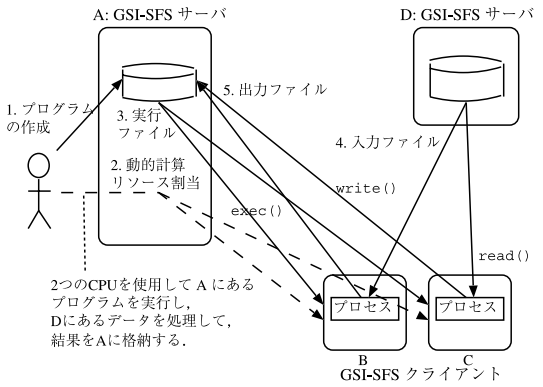

図 4 GSI-SFS のユースケース
Fig. 4 A use case of the GSI-SFS.

管理コストが高くなるという問題もある．

4.2 利 便 性

GSI-SFS は科学技術計算における科学者の負担の軽減を主な目的として開発された．本節では科学技術計算における GSI-SFS のユースケースを示し，利便性について評価する．ユースケースとしては科学者がプログラムを作成し，グリッド上のデータを複数の CPU を使用して処理するという典型的な場合を考える（**図 4**）．

SFS は分散ファイルシステムであるため，プログラムの作成においてユーザは POSIX API を使用することができる（図 4–1）．このことはグリッドの専門知識を持たないユーザの負担軽減につながる．また，既存の多くのプログラムをそのまま利用することもできる．

グリッドではホストのアーキテクチャや性能などのリソース情報に基づいて，動的にプログラムを実行するホストを決定することがしばしば行われる（図 4–2）．GSI-SFS でも SFS と同様にグローバルファイルシステムイメージを実現しており，どのホストでプログラムが実行されても同じファイル名でグリッド上のファイルを特定できる．ユーザはプログラムがどのホストで実行されるかを意識せずに実行ファイル（図 4–3），入力ファイル（図 4–4），出力ファイル（図 4–5）を指定することができる．SFS では実行ファイルを実行するホストにコピーすることなく直接起動することもできる．

すべてのマウントはアクセス時にオンデマンドで行われ，ユーザは使用される可能性のあるすべてのホストをあらかじめマウントしておく必要はない．マウント時には GSI の認証が行われるが，このユースケースでは委譲証明書が利用できるためユーザが認証を意識する必要はなく，シングルサインオンが実現されている．

このように GSI-SFS は利便性の高いファイルアクセスを提供するが，グリッドのデータ管理における最も低いレベルのデータアクセス機能を提供するに過ぎない．大規模なシステムの構築にはより上位のメタデータカタログサービスやレプリケーションサービスなどの併用が必要となる．

4.3 SFS のスループット

GSI-SFS を SFS をベースに開発したのは，設計時にパフォーマンスよりも利便性とセキュリティを重視したためである．しかし，GSI-SFS は主に科学技術計算を対象としているため，パフォーマンスについても検証しておく必要があると考えた．科学技術計算においては特にスループットが重要であるため，本節では SFS のスループットを評価する．

SFS は UDP の NFS を TCP で中継する．広帯域かつ遅延の大きいネットワークにおいては TCP の転送効率が低下することが知られている．また，NFS は遅延の小さい LAN を前提に設計され RPC を使用するため，遅延の大きいネットワークにおいてはスループットが低下すると予測される．そこで SFS，TCP，NFS のスループットを測定した．また，比較対象として GridFTP についても測定を行った．測定では**表 2** に示す同じ仕様の 2 台の PC をハブを介して接続し，Linux 用の WAN エミュレータである NISTNet☆を使用して LAN 上で帯域と遅延をエミュレートした．TCP の設定は Linux のデフォルト値を使用しており，受信バッファサイズの最小/標準/最大はそれぞれ 4,096/87,380/174,760 B，同様に送信バッファサイズは 4,096/16,384/131,072 B，ウインドウスケーリングは有効である．

☆ http://snad.ncsl.nist.gov/itg/nistnet/

表 2 測定に使用した PC の仕様
Table 2 Specification of the PCs.

CPU	Intel Xeon 2.8 GHz × 2
Memory	2 GB
Disk	Maxtor 4A250J0
Network	Intel 82545EM (64-bit PCI)
RedHat Linux	9 (kernel 2.4.20)
Globus Toolkit	2.4.3
SFS	0.7.2
GSI-SFS	0.0.7
NISTNet	2.0.12
Netperf	2.2pl2

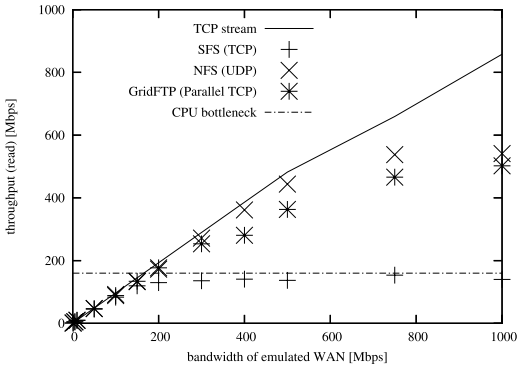

図 5 帯域と読み取りスループット
Fig. 5 Bandwidth vs. reading throughput.

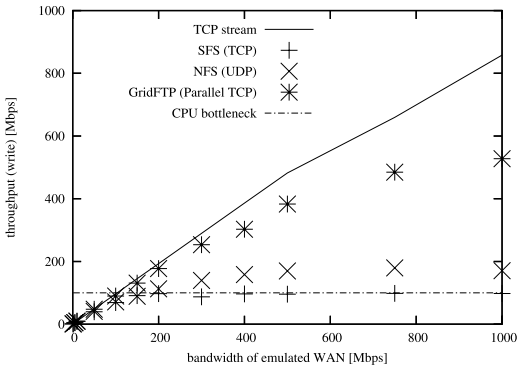

図 6 帯域と書き込みスループット
Fig. 6 Bandwidth vs. writing throughput.

まず NISTNet で往復遅延（Round-trip Time: RTT）を 0 秒に固定し帯域を変化させ，内容が乱数で埋められた 100 MB のバイナリファイルのコピーに要する時間を計測しスループットを計算した．GridFTP ではコネクション数 8 を指定した．NFS には UDP を使用した．サーバからクライアントにコピーしたときの結果を**図 5** に，クライアントからサーバにコピーしたときの結果を**図 6** に示す．なお，TCP ストリームのスループットはネットワークベンチマークツール

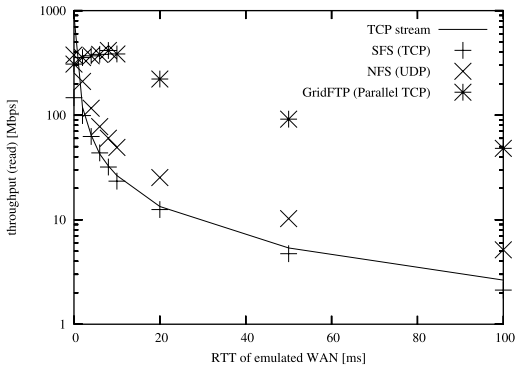

図 7 RTT と読み取りスループット
Fig. 7 RTT vs. reading throughput.

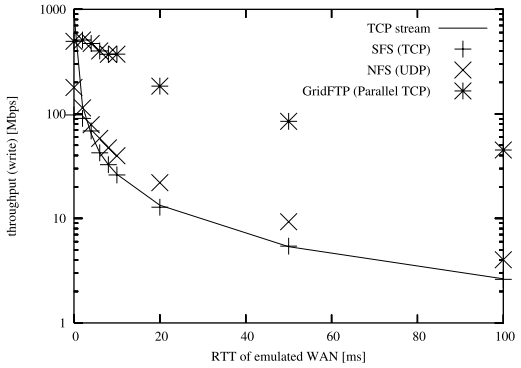

図 8 RTT と書き込みスループット
Fig. 8 RTT vs. writing throughput.

Netperf☆で測定した値である．図 5 では 160 Mbps 程度，図 6 では 100 Mbps 程度で SFS のスループットは飽和している．一方で NFS や TCP のスループットは飽和しておらず，SFS では暗号化処理などを行う CPU がボトルネックとなっていると考えらえる．

次に帯域を 1000 Mbps に固定し，RTT を変化させて同様の計測を行った．サーバからクライアントにコピーしたときの結果を**図 7** に，クライアントからサーバにコピーしたときの結果を**図 8** に示す．図 7 と図 8 の両方において RTT が大きくなるに従って TCP ストリームと NFS のスループットは低下している．先に述べたように，これは TCP と NFS のプロトコル上の問題であると考えられる．SFS は NFS を TCP で中継する仕組みであるため，SFS のスループットは NFS と TCP を超えることができない．

以上より，SFS のスループットはネットワークの帯域よりも遅延に大きく影響され，遅延の大きいネットワークではスループットが低下することが分かった．

☆ http://www.netperf.org/netperf/NetperfPage.html

文献 14) では，いくつかの科学技術アプリケーションの消費するリソースが示されているが，10 kbps から 10 Mbps 程度までの I/O トラフィックを発生させるものとして，SETI@home，バイオインフォマティクスの BLAST[15]，地球環境のシミュレーション IBIS，高エネルギー物理の CMS，分子動力学シミュレーション Nautilus，天体物理の AMANDA があげられている．GSI-SFS はこのような CPU への負荷が大きく，高速なデータアクセスはさほど必要とされないアプリケーション，そして実行ファイルや設定ファイルの転送などに適していると考える．

5. 応用事例

現在，大阪大学と中国の中国科学院ではグリッド技術を用いて，互いの生物情報データベースやプロセッサを共有するシステムの構築を行っている[16]．中国は多数の固有種の生物データベースを保有しており，このデータベースは世界中の生物学者から注目されている．また，大阪大学には高性能 PC クラスタなどで構成された「バイオグリッド基盤システム」が導入されており，高速な解析演算が可能である．これらのデータリソースと計算リソースをグリッドで接続することで生物学の発展が期待され，またこれを応用して in-silico 創薬を行う研究開発も進められている．

このプロジェクトにおいて，アプリケーションからのデータアクセスに GSI-SFS を使用した．システムの構成を**図 9** に示す．主なユーザは生物学者や製薬会社の研究員である．ユーザは従来から BLAST や ClustalW[17] と呼ばれる解析アプリケーションを利用していた．これらのユーザはグリッドに関する専門知識を持たず，アプリケーションをグリッドのために拡張することは困難であった．そこで GSI-SFS を使用することでこれらのアプリケーションを拡張することなく，大阪大学のクラスタを用いた中国科学院にある生物データの解析を実現した．

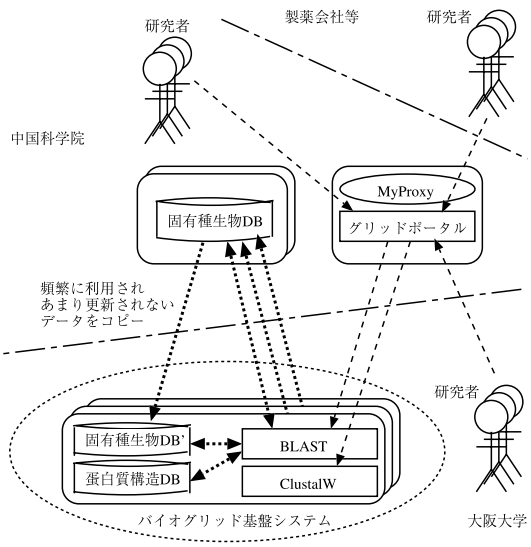

図 9 生物情報を対象としたグリッドテストベッド
Fig. 9 A grid testbed for bioinformatics.

ユーザは中国科学院に設置されているグリッドポータルから双方のデータを利用して解析を行うことができる．グリッドポータルではシングルサインオンを実現するために MyProxy[18] を使用している．ユーザは大阪大学および中国科学院の多数のホストを利用するが，MyProxy にプロキシ証明書を格納しておくことで認証を意識することなく利用できる．

中国科学院に配置されたあるデータベースから大阪大学の 1 ノードで BLAST の検索処理を行った場合，検索条件やネットワーク状態にも依存するが 1 時間程度を要する．この処理では BLAST のプログラムが約 500 MB を読み出し，書き込みは行わない．現状では大阪大学と中国科学院との間のネットワーク帯域が狭く TCP ストリームのスループットは 1 Mbps 程度で，20 以上のルータを経由しており RTT は 400 ミリ秒以上であった．同じ処理を 32 ノードで分散して行った場合，1 つの GSI-SFS サーバに TCP のコネクションが 32 本張られるため，GridFTP の並列転送機能と同様の効果が得られ 30 分程度で処理が完了した．頻繁に利用され，あまり更新されないデータについては大阪大学のクラスタの全ノードに手作業でコピー（レプリケーション）されている．大阪大学のデータを指定した場合，先に述べた検索処理を 1 ノードで 1 分，32 ノードで 30 秒程度で処理することができる．今後はレプリケーションサービスを導入してレプリケーションを自動化，効率化し，より有用なシステムに発展させることを検討している．ClustalW については入出力が非常に少なく，また計算量も少ないため高性能なコンピュータを利用する必要はないが，ユーザの利便性を考慮して BLAST と連携できるようにしている．

創薬においては非常に高い機密性が要求される．たとえ公開データにアクセスする際においても，どの製薬会社がどのデータにアクセスしているかという情報さえも機密とされる．SFS では，ネットワークを流れるデータは暗号化，検証される．また，ホストを共有した場合においても，ユーザが他のユーザがどのホストのどのファイルにアクセスしているか知ることはできない．

6. 今後の方針

グリッドでは多数の組織のユーザがリソースを共有するため柔軟なアクセス制御が求められるが，GSI-SFS ではこれが実現されていない．この問題を解決するために，証明書の識別名に基づいてユーザやグループごとに明示的なアクセス権限を設定できる Access Control List（ACL）を実装することを検討している．そして，ホストごとに `grid-mapfile` や ACL を設定する管理コストの増大に対応するために，Community Authorization Service（CAS）[19] への対応も検討している．CAS ではコミュニティ全体でセキュリティポリシを共有でき，管理コストの軽減が期待できる．

現在 GGF の BOF Grid File System Working Group（GFS-WG）ではグリッドのファイルシステムに関して議論が行われている．GFS-WG はグリッドファイルシステムに必要とされる機能として階層的な仮想名前空間，柔軟なアクセス制御，メタデータの管理などを提案しているが，GSI-SFS このような高度な機能を実装するに至っていない．またすでに GSI-SFS を利用しているユーザからは，エンディアンの自動変換機能などアーキテクチャの違いを吸収する API への要望もある．今後も GFS-WG の提案やユーザの要望を参考しながら GSI-SFS をより有用なものへと拡張していくことを考えている．そして SFS にグリッドに適した変更，修正を行うことで性能を改善し，高いスループットが求められるアプリケーションや，高いスケーラビリティが必要とされるグリッド環境にも対応していきたいと考えている．

7. ま と め

本論文では GSI を使用して SFS を拡張した分散ファイルシステム GSI-SFS を提案した．GSI-SFS は GSI の認証を SFS にも適用することで，グリッド環境に SFS をシームレスに統合する．GSI-SFS ではシングルサインオン，グローバルファイルシステムイメージ，オンデマンドなマウントなど高い利便性を実現しながらもセキュリティの低下は最小限にとどめている．GSI-SFS は利便性とセキュリティ強度の高さから 5 章で紹介したようなプロジェクトでも利用され，そのシングルサインオン機能の有効性が実証された．

今後の課題としては，まずアクセス制御の柔軟性を向上させることがあげられる．そして，ユーザの要望や GGF の GFS-WG の提案を参考にしながら GSI-SFS をより有用なものにできるよう，機能拡張と性能向上を行っていきたいと考えている．現在，GSI-SFS のソースコードやドキュメントは Biogrid project のウェブサイト☆で公開している．

謝辞 本研究は文部科学省科学技術振興費主要 5 分野の研究開発委託事業の IT プログラム「スーパーコンピュータネットワークの構築」の一環として実施された研究成果の一部である．

本ファイルシステムの開発および評価においてご協力をいただきました中国科学院微生物研究所の馬俊才主任，秦野彰二氏，三井情報開発（株）バイオサイエンス本部の木戸善之氏に感謝いたします．

本論文を査読し，多数の有益なコメントをくださいました査読者の方々にこの場を借りて感謝いたします．

参 考 文 献

1) Foster, I., Kesselman, C. and Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *International Journal of Supercomputer Applications*, Vol.15, No.3, pp.200–222 (2001).
2) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications*, Vol.11, No.2, pp.115–128 (1997).
3) Allcock, B., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnal, D. and Tuecke, S.: Data Management and Transfer in High Performance Computational Grid Environments, *Parallel Computing Journal*, Vol.28, No.5, pp.749–771 (2002).
4) Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Liming, L., Meder, S. and Tuecke, S.: GridFTP Protocol Specification, *GridFTP Working Group Document*, Global Grid Forum (2002).
5) Horowitz, M. and Lunt, S.: FTP Security Extensions, Request for Comments 2228, Network Working Group, Internet Engineering Task Force (1997).
6) Butler, R., Engert, D., Foster, I., Kesselman, C., Tuecke, S., Volmer, J. and Welch, V.: A National-scale Authentication Infrastructure, *IEEE Computer*, Vol.33, No.12, pp.60–66 (2000).
7) Stockinger, H., Samar, A., Allcock, B., Foster, I., Holtman, K. and Tierney, B.: File and Object Replication in Data Grids, *Journal of Cluster Computing*, Vol.5, No.3, pp.305–314 (2003).
8) Thain, D. and Livny, M.: Parrot: Transparent User-Level Middleware for Data-Intensive

☆ http://www.biogrid.jp/

Computing, *Workshop on Adaptive Grid Middleware* (2003).
9) Rajasekar, A., Wan, M. and Moore, R.: MySRB & SRB—Components of a Data Grid, *The 11th International Symposium on High Performance Distributed Computing*, pp.301–310 (2002).
10) Tatebe, O., Morita, Y., Matsuoka, S., Soda, N. and Sekiguchi, S.: Grid Datafarm Architecture for Petascale Data Intensive Computing, *The 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp.102–110 (2002).
11) Morris, J.H., Satyanarayanan, M., Conner, M.H., Howard, J.H., Rosenthal, D.S. and Smith, F.D.: Andrew: A Distributed Personal Computing Environment, *Comm.ACM*, Vol.29, No.3, pp.184–201 (1986).
12) Satyanarayanan, M., Kistler, J.J. and Siegel, E.H.: Coda: A Resilient Distributed File System, *IEEE Workshop on Workstation Operating Systems* (1987).
13) Mazières, D.: Self-certifying File System, Ph.D. Thesis, Massachusetts Institute of Technology, Boston (2000).
14) Thain, D., Bent, J., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H. and Livny, M.: Pipeline and Batch Sharing in Grid Workloads, *The 12th International Symposium on High Performance Distributed Computing* (2003).
15) Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J.: Basic Local Alignment Search Tool, *Journal of Molecular Biology*, Vol.215, pp.403–410 (1990).
16) Biogrid project: The Collaboration for the Grid Research by Bioresearches in Osaka University and Chinese Academic of Sciences, Press release, Cybermedia Center, Osaka University and Information Network Center, Institute of Microbiology, Chinese Academy of Sciences (2003). http://www.biogrid.jp/e/archive/asia/index.html
17) Thompson, J.D., Higgins, D.G. and Gibson, T.J.: CLUSTAL W: Improving the Sensitivity of Progressive Multiple Sequence Alignment Through Sequence Weighting, Position-specific Gap Penalties and Weight Matrix Choice, *Nucleic Acids Research*, Vol.22, pp.4673–4680 (1994).
18) Novotny, J., Tuecke, S. and Welch, V.: An Online Credential Repository for the Grid: MyProxy, *The 10th International Symposium on High Performance Distributed Computing* (2001).
19) Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S.: A Community Authorization Service for Group Collaboration, *The IEEE 3rd International Workshop on Policies for Distributed Systems and Networks* (2002).

(平成 15 年 10 月 9 日受付)
(平成 16 年 2 月 6 日採録)

武田 伸悟（学生会員）
1980 年生．2003 年大阪大学工学部電子情報エネルギー工学科卒業．現在大阪大学大学院情報科学研究科博士前期課程在学中．

伊達 進（正会員）
1975 年生．2000 年大阪大学大学院基礎工学研究科物理系専攻博士前期課程修了．2002 年大阪大学大学院工学研究科システム工学専攻博士後期課程修了．工学博士．2002 年大阪大学大学院情報科学研究科バイオ情報工学専攻助手．現在に至る．グリッド技術のバイオ，メディカルへの応用とその基礎技術の研究を行う．

下條 真司（正会員）
1958 年生．1986 年大阪大学大学院基礎工学研究科物理系専攻博士後期課程修了．工学博士．1986 年大阪大学基礎工学部助手．1989 年大阪大学大型計算機センター講師．1991 年大阪大学大型計算機センター助教授．この間（1996 年 2 月から 9 月まで），米国カリフォルニア大学アーバイン校客員研究員．1998 年大阪大学大型計算機センター教授．2000 年大阪大学サイバーメディアセンター教授（副センター長），現在に至る．