グリッド認証基盤に適応した
ファイルアクセスとモニタリングに関する研究

2008年1月

武田　伸悟

グリッド認証基盤に適応した
ファイルアクセスとモニタリングに関する研究

提出先　　大阪大学大学院情報科学研究科

提出年月　2008年1月

武田　伸悟

研究業績

1. 学術論文誌発表論文

1. 武田伸悟, 伊達進, 下條真司, "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," 情報処理学会論文誌コンピューティングシステム, vol. 45, no. SIG6 (ACS 6), pp. 223–233, May 2004.

2. 研究会等発表論文（査読付）

1. 武田伸悟, 伊達進, J. Zhang, B. S. Lee, 下條真司, "グリッドにおけるサービス要求フローに着目したセキュリティモニタリング," 第 19 回 コンピュータシステム・シンポジウム (ComSys 2007), vol. 2007, pp. 209–216, November 2007.
2. S. Takeda, S. Date, J. Zhang, B. S. Lee, and S. Shimojo, "Security monitoring extension for MOGAS," in *Proceedings of the 3rd International Workshop on Grid Computing & Applications*, pp. 128–137, June 2007.
3. T. Tashiro, S. Date, S. Takeda, I. Hasegawa, and S. Shimojo, "Architecture of authorization mechanism for medical data sharing on the Grid," in *Proceedings of HealthGrid 2006*, pp. 358–367, June 2006.
4. T. Tashiro, S. Date, S. Takeda, I. Hasegawa, and S. Shimojo, "Practice and experience of building a medical application with PERMIS-based access control mechanism," in *Proceedings CD-ROM of the 6th IEEE International Conference on Computer and Information Technology (CIT2006)*, September 2006.
5. Y. Kido, S. Date, S. Takeda, S. Hatano, J. Ma, S. Shimojo, and H. Matsuda, "Architecture of a Grid-enabled research platform with location-transparency for bioinformatics," in *Proceedings of the 15th International Conference on Genome Informatics (Genome Informatics 2004)*, vol. 15, pp. 3–12, December 2004.

3. その他の研究会等発表論文

1. 武田伸悟, 伊達進, 下條真司, “グリッドにおける大規模ファイル共有システムの構築,” 情報処理学会研究報告 2003-HPC-95, pp. 167–172, August 2003.
2. 武田伸悟, 伊達進, 下條真司, “グリッドファイルシステム GSI-SFS,” 情報処理学会研究報告 2003-OS-93, pp. 97–104, May 2003.
3. 史宏宇, 武田伸悟, 長谷川一郎, 伊達進, 水野（松本）由子, 下條真司, “IPv6 におけるセキュアなグリッド環境の構築,” 情報処理学会研究報告 2003-HPC-94, pp. 25–30, June 2003.

内容梗概

本論文は，筆者が 2003 年から現在までに，大阪大学大学院情報科学研究科マルチメディア工学専攻応用メディア工学講座在学中に行った研究の成果をまとめたものである．

近年，学際領域における共同研究を円滑に推進する環境として，グリッドが注目されている．グリッドは，地理的に分散した複数の組織間で，コンピュータの計算能力やストレージ，プログラム，データなどのリソースを，それらが分散していることをユーザが意識せずに共有できる環境である．例えば，生体メカニズムの解明や新薬の開発のためには，医学，生物学，物理学，情報学など様々な分野の専門家らが，シミュレーションプログラム，分子構造データ，高速コンピュータなどの共有を行っている．グリッドでは共通の認証基盤の上でプログラム実行やデータ転送のサービスが提供され，ユーザはシングルサインオンでリソースを利用することができる．この機能により，単一組織のリソースでは行えない複雑な科学技術計算が可能になる．しかし，グリッド認証基盤の上にユーザが計算プログラムやデータを安全に配置すること，また管理者が安全にグリッドを運用することには，依然多大な労力を伴い困難である．そのため，今日ではユーザ及び管理者の両視点でこれらの負担を軽減するグリッド技術が求められている．このような背景から，本研究ではグリッド認証基盤に適応したファイルアクセス及びモニタリングに着目し，安全なグリッドを構築する技術に関する研究開発を行う．

本論文では，まずユーザ及び管理者の視点でグリッド利用を困難にする要因の分析を行い，ユーザの視点から分散データへのアクセス，管理者の視点から認証エラーなどのセキュリティ異常のモニタリングという点について着目する．そして技術課題として，位置透過性の高い遠隔ファイルアクセスと，グリッド認証基盤の効率的なモニタリングの実現があることを述べる．

位置透過性の高い遠隔ファイルアクセスの実現のために，オペレーティングシステムのカーネル領域で動作することによって高い位置透過性を実現している分散ファイルシステムを，グリッド認証基盤に適応させる手法を提案する．これにより，ユーザはシングルサインオンで位置透過的にグリッド上のファイルにアクセスでき，グリッド上への計算プログラムやデータの安全かつ容易な配置が可能となる．本システムを用いて，分散した生物

データを高性能クラスタで解析するグリッドを構築し，従来のファイル転送手法を使用するよりも効率的に解析できることを確認する．

グリッド認証基盤の効率的なモニタリングの実現のためには，グリッド認証基盤から情報を収集し，不正利用や機密漏えいと関連しうる，グリッド特有の複数組織にまたがった異常の早期発見と原因分析を統合的に支援するセキュリティモニタリングシステムを提案する．これにより，管理者の安全なグリッド運用を容易にする．この実証のため，本システムを学術機関を主としたグリッドテストベッドに展開して評価実験を行い，従来のモニタリングシステムでは発見が困難な異常を短時間で発見できることを検証する．

目次

第 1 章

序論

1.1 背景

学際領域における共同研究を円滑に推進する環境として，Ian Foster らが提唱したグリッド [1-3] が注目されている．グリッドは，地理的に分散した複数の組織間で，インターネット上に分散するコンピュータの計算能力やストレージ，プログラム，データなどのリソースを共有する環境である．例えば，生体メカニズムの解明や新薬の開発のためには，医学，生物学，物理学，情報学など様々な分野の専門家らが，シミュレーションプログラム，分子構造データ，高速コンピュータなどの共有を行っている．グリッドにおける計算能力の共有では，遊休リソースを他の組織が利用できるようにすることでリソースを有効活用でき，低コストで大きな計算能力を得ることができる．これにより，従来組織内のリソースだけでは難しかった大規模かつ複雑な計算が可能となり，研究効率が向上することが期待されている．薬の候補となる化合物を絞り込むスクリーニングはグリッドでの並列分散処理に適したアプリケーションの一例であり，大規模な化合物データベースをグリッドで高速にスクリーニングする研究が報告されている [4, 5]．また，高解像度天体望遠鏡などの近年の高度計測機器は非常に大きなデータを出力し，このような大きなデータを高速転送，蓄積，並列処理，可視化することにもグリッド技術の応用が進められている [6]．

グリッドでは複雑な科学技術計算を短時間で行うことを目的とし，高性能のコンピュータがインターネットに高速ネットワークで常時接続される．特に，学術機関を結ぶ実験ネットワークは高速で，中には 10Gbps を超えるものもあり [7]，膨大なデータを短時間で転送することができる．このような高速ネットワークに接続された高性能コンピュータが，Distributed Denial of Service（DDoS：分散サービス拒否）攻撃や SPAM の送信を行う悪意を持った第三者に侵入され，不正利用された場合にはネットワークシステムが機能停止に陥るなど，非常に大きな被害が発生することが懸念される．

同時に，グリッド技術の応用分野が広がるにつれてデータ機密保護への対策の重要性が高まっており，必要とされるアプリケーションが増加している．文献 [8] では，IPv6 と IPsec を応用し，グリッドにおいてネットワークレベルでデータ機密保護を行う試みについて報告している．文献 [9, 10] では，グリッド技術を遠隔医療に応用する試みについて報告している．この例では，病院のデータベースに蓄積されている電子カルテに含まれる情報を，患者の疾患に関する専門知識を持つ他病院の医師がアクセスする．電子カルテには患者の個人情報が含まれているため，診察に携わるユーザ以外からアクセスできないように機密保護が必要である．このように，グリッドにおけるセキュリティの重要性が高まってきており，グリッドを安全に運用，管理することは重要な課題となっている．

グリッドのような分散システムを安全に運用するためには，認証（Authentication），認可（Authorization），アカウンティング（Accounting）の 3 つの要素が重要である．これらを備えた構成は英語のアルファベットの頭文字を取って AAA アーキテクチャ [11] として提唱されている．認証はユーザ ID とパスワードの組などでユーザやリソースを識別することである．認可は認証の結果に基づいてユーザの操作を許可するか拒否するかを判断し，アクセスを制御することである．アカウンティングはユーザの利用履歴を管理し，課金などの目的に利用することである．これら 3 つの要素の中で，認証は認可やアカウンティングの前提となる，最も基本で重要な要素である．

グリッドでは，ユーザは複数組織に分散した多数のリソースを利用する．このため，リソースを利用するごとにパスワード入力などの認証操作をユーザに要求することは研究の効率化を妨げることになり，グリッドにおいては一度の認証操作で複数のサービスを利用できること，すなわち，シングルサインオンが不可欠である．グリッドではシングルサインオンの実現のために，公開鍵暗号認証技術を採用した共通の基盤の上で各種サービスが提供される．本研究では，この基盤をグリッド認証基盤と呼ぶ．グリッド認証基盤の上で，データ転送，リソース管理などのサービスを提供することで，ユーザは認証操作を意識することなくシングルサインオンで複数のリソースが利用できる．

しかし，グリッドのユーザである様々な科学技術分野の専門家らにとって，グリッド認証基盤の上で科学技術計算を行うことは敷居が高いのが実情である．上述したシングルサインオン機能を利用するためには，ユーザはこれまでに作成してきた科学技術計算プログラムを拡張し，グリッド認証基盤に適応させなくてはならない．グリッド認証基盤では公開鍵暗号を応用した複雑な手法が用いられており情報科学を専門としないユーザは習得に時間を要するだけでなく，また提供されているライブラリを用いてプログラムを拡張することにもグリッドのソフトウェア仕様に関する詳細な知識が要求される．さらに，拡張した計算プログラムとその計算に必要なデータは，あらかじめ使用する計算リソース上に配置しておかなければシングルサインオンでの計算が行えない．ユーザが使用されうる計算リソースを全て把握することは難しく，またそれらの計算リソースにプログラムとデータ

を配置することには多大な労力と時間を要することがある．このように，ユーザにとってシングルサインオンで利用できる計算環境を整えることは困難であり，グリッド利用による研究の効率化の妨げとなっている．

一方，リソースを管理している各組織の管理者が，グリッド認証基盤を安全に運用することにも多大な労力が伴うのが実情である．グリッドを安全に運用するためには，管理者はグリッド上でどのような処理がなされているかをモニタリングし，不正利用や機密漏えいの前兆が発見された場合は迅速に対応して被害の発生を防止しなければならない．しかし，シングルサインオンは，ユーザによる複数組織に分散した計算リソースの利用利便性を高める一方で，管理者によるグリッドのセキュリティ状態の把握を困難にしている．セキュリティ状態の把握のためには，管理者はリソース上に出力されているログ情報を閲覧し，他のリソースのログ情報や他組織のネットワークの情報なども参照しながら情報分析する必要がある．日常的にこのような作業を行うことには多大な労力と時間を要し，このことがグリッドの安全な運用を行う上での障壁となっている．

1.2 本研究の目的と課題

グリッド技術によって効率的な科学技術計算を実現するためには，先に述べたユーザと管理者の両方の問題を解決し，各分野の専門家らが専門とする問題領域に集中できるようにすることが肝要である．本研究では，このような観点から，グリッド技術を用いた科学技術計算におけるユーザと管理者の負担低減を目的とし，利便性と安全性を両立するグリッド技術の研究開発を行う．本目的のために，本論文ではユーザのプログラムとデータの配置に関する問題，及び，管理者のセキュリティ状態の把握に関する問題に着目する．

ユーザは，Fortran や C 言語などで，多くの科学技術計算プログラムをすでに作成している．研究の効率化のためには，これらのプログラムと計算に必要なデータをグリッド上の各リソースに，ユーザが安全かつ簡便に配置できなければならない．リソース上で実行されるプログラムは不正利用を防ぐために整合性を確保する必要があり，また計算に使用されるデータは個人情報や企業秘密の漏えいを防ぐために機密性も確保する必要がある．これらの理由から，安全性を犠牲にすることなく，科学技術計算プログラムとデータのグリッド展開を容易にする技術の実現が課題となっている．

管理者は，グリッドを安全に運用するために，リソースをモニタリングして不正アクセスや機密漏えいの兆候を迅速に発見しなければならない．このためには，セキュリティの観点からグリッドをモニタリングする技術が必要である．管理者が，それらの兆候を発見した際には，すばやく原因を追及し対応することも重要である．そこで，管理者の異常の発見と分析を統合的に支援するモニタリングシステムを実現することが課題となっている．

1.3 論文構成

2章では，グリッドを構築する上でユーザと管理者の問題となっている点をより詳細に分析する．管理者は様々なソフトウェアを組み合わせてグリッドを構築する．ここでは，まずそれらのソフトウェアで使用されている技術を調査し，安全性と利便性について評価する．その後，ユーザと管理者の負担を低減するための技術的課題について論じる．

3章では，ユーザの負担を低減するために，ファイルの位置を意識させない位置透過的なファイルアクセスを実現する手法を提案する [12-14]．本研究では，分散ファイルシステムの高い位置透過性に着目し，既存の分散ファイルシステムをグリッド認証基盤に適応させることで，安全性と利便性を両立する．グリッド認証基盤に適応させシングルサインオンを実現するためには，分散ファイルシステムの利用に登録が必要な対称鍵のペアを，グリッド認証基盤の機能を応用して自動的かつ安全に登録するメカニズムを提案する．また，計算リソース上に生成される平文の秘密鍵が漏えいした際のリスクを低減するために，一般的な手法である証明書の有効期限による制限に加えて，権限委譲によって利用できるサーバを制限する手法を提案する．

4章では，管理者の負担を低減するために，セキュリティの観点からリソースをモニタリングするシステムを提案する [15, 16]．提案システムは，グリッド特有の複数組織にまたがる不正利用と関連しうる異常の発見とその原因分析にかかる管理者の負担を低減するために，複数組織のリソースに散在する数万行のグリッド認証基盤のログファイルに出力される認証と認可の情報を収集，蓄積，可視化し，管理者らが視覚的に異常発見と原因分析を行うことを可能とする．複数組織にまたがる各リソース上でのログファイルからの情報取得は，本システムの導入にかかる管理者の負担を小さくするために，オペレーティングシステムに依存せずコンパイルが不要なスクリプトで実現する．情報の送信にはHTTPSのPOSTメソッドを使用し，安全性と互換性を両立する．取得した情報は各リソースから中央データベースに送信して集中的に管理することで，運用にかかる組織の管理者の負担が増大しないよう配慮する．蓄積された情報を総合的に解析し，複数組織にまたがる事象を管理者が容易に発見できるよう，グラフと表を用いて異常を抽出する手法をWebインタフェース上に実現する．このWebインターフェースには管理ポリシの異なる複数の組織の管理者が安全に情報を共有するためのアクセス制御機構を組み込む．

5章では，本研究にて得られた成果を要約し，今後の課題を述べる．

第２章

グリッドセキュリティの現状と課題

2.1　まえがき

グリッド技術を用いた科学技術計算が注目を集めるとともに，様々なグリッドのサービスが世界中のグリッド研究者らによって開発されている．グリッド認証基盤は共通の認証機能をサービスに提供することで，これらのサービスをユーザがシングルサインオンで利用することを可能にする．グリッド認証基盤の上で各種サービスを提供することで，ユーザは認証操作を意識することなく分散したリソースを利用できる．しかし，1 章で記したように，グリッドに関する専門知識を有さないユーザがグリッド認証基盤上にプログラムとデータを展開することは困難であるのが現状である．また，管理者にとっても，グリッドでは多数のリソース要求が発生するため，セキュリティ状態を把握することには多大な労力が伴う．これらの理由から，ユーザと管理者の両視点で安全性と利便性を両立するセキュリティ技術の開発が課題となっている．

グリッド構築のためには，グリッド認証基盤に加え様々な技術を応用したソフトウェアが組み合わせて用いられる．科学技術計算の並列分散処理では，遠隔プログラム実行，データ転送，リソース管理，またそれらを利用するためのインタフェースが必要であり，管理者はそれぞれの機能を提供するソフトウェアの中から目的に適したものを選択して導入する．このため，安全性を犠牲にすることなくユーザと管理者の利便性を改善するためには，グリッド認証基盤だけでなく，組み合わせて用いられる技術についても調査を行い，グリッドシステム全体について考察する必要がある．

本章では，一般的なグリッドにおける科学技術計算で利用される技術を調査し，それらの技術を応用したソフトウェアを組み合わせてグリッドを構築する際の，ユーザと管理者の安全性と利便性の両立を困難にしている原因を分析する．そして，1 章で掲げた課題をより詳細にし，科学技術計算の効率化に向けてのユーザと管理者のグリッドセキュリティ技術に対する要求を明確にする．

以下に本章の構成を示す．2.2 節では，まず本研究で着目するグリッド認証基盤とその実装について調査し，シングルサインオンの仕組みを説明する．2.3 節では，グリッド認証基盤上で科学技術計算を行うために導入されるソフトウェアを調査する．そして，グリッド認証基盤とそれらを組み合わせ，どのようにシングルサインオンの計算環境が実現されるかを説明する．2.4 節では，2.2 節と 2.3 節をふまえて，ユーザがグリッド認証基盤上で科学技術計算を行う上での問題点，及び，管理者がグリッドのセキュリティ状態を把握する上での問題点を挙げ，それぞれを解決するための課題を述べる．

2.2 グリッド認証基盤

本論文では，グリッドにおける認証に必要な機能を提供するソフトウェアをグリッド認証基盤と定義する．グリッド認証基盤は，ユーザや管理者が証明書を操作するための機能や，サービスがユーザを認証するための機能を提供する．本節では，まずグリッドによる複数組織間でのリソース共有の概要を示し，次に以後の議論で重要である各種証明書とそれを用いた認証の手順，及び，グリッド認証基盤の具体的な実装について説明する．

2.2.1 グリッドにおける複数組織間でのリソース共有

グリッドは地理的に分散した複数組織間でネットワークを介して，計算能力，ストレージ，プログラム，データなど多岐にわたるリソースを共有する環境である．複数組織間でリソースを共有する例を図 2.1 に示す．この例では，病院，製薬会社，大学の 3 つの組織がグリッドを構成しており，その中の様々な分野の専門家らが計算リソース，データ，プログラムを共有している．このようなリソース共有が可能になることで，製薬会社の研究者が病院の疾患情報を参照し，大学の物理学者によって開発された分子計算プログラムを用いて，大学と製薬会社の両方の計算リソースを使用して並列計算を行えるなど，科学技術研究の効率化が期待される．

グリッド技術は数十の組織が参加し，数百のユーザが数百のリソースを共有する大規模な環境まで想定して研究されている．このような大規模なグリッドにおいて円滑なリソース共有を実現するためには拡張性の高い認証手段が必要不可欠である．グリッドでは拡張性を確保するために公開鍵基盤（Public Key Infrastructure：PKI） [17] を基礎とした認証方式が用いられる．PKI では信頼された Certificate Authority（CA：認証局）がユーザとリソースに電子証明書を発行し，ユーザとリソースは互いの証明書を検証しあうことで相互認証を行う．さらにグリッドでは，多数のリソースを円滑に利用可能とするために PKI を応用した特有の技術によってシングルサインオンが実現される．以下では，利便性と安全性の両立を困難にしている原因を分析するために重要である，PKI を応用したシングル

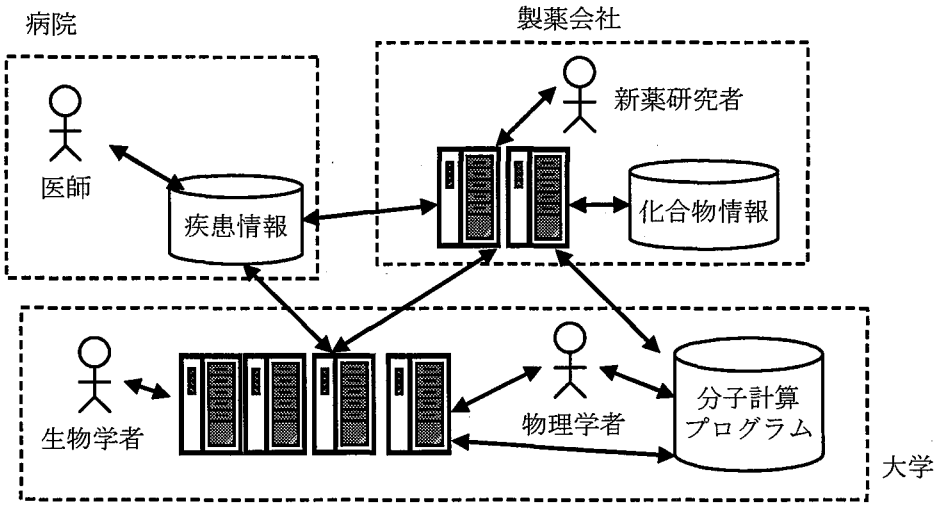

図 2.1 複数組織間でのリソース共有の例.

サインオンの原理と，それを実装した具体的なミドルウェアについて説明する.

2.2.2 グリッドで使用される標準証明書

グリッド認証基盤では，ユーザとサービスの識別のために X.509 証明書 [17] が使用される．グリッドのユーザは CA から発行された X.509 形式のユーザ証明書を持つ．図 2.2 にユーザ証明書の例を示す．図中の Issuer の項目は，このユーザ証明書を発行した CA を表している．カンマで区切られたそれぞれの部分は，Country（国）が JP（日本），Organization（組織）が Osaka University，Organization Unit（組織内の部局）が Cybermedia Center，Common Name（名称）が Biogrid CA であることを表している．この CA は

```
/C=JP/O=Osaka University/OU=Cybermedia Center/CN=Biogrid CA
```

と表すことができ，これは Distinguished Name（DN：識別名）と呼ばれる．図中の Subject の項目は，このユーザ証明書で識別されるユーザを表している．CA と同様に，このユーザは識別名で

```
/C=JP/O=Osaka University/OU=Cybermedia Center/CN=Shingo Takeda
```

と表される．図中の Validity の項目は，このユーザ証明書の期限を表しており，有効期間は 1 年である．図中の RSA Public Key の項目は，ユーザの公開鍵であり，認証や暗号通信路の確立に使用される．これに対応する秘密鍵はユーザのクライアントに暗号化された状態で保存されておりネットワークに送信されることはない.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 8 (0x8)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=JP, O=Osaka University, OU=Cybermedia Center,
CN=Biogrid CA
        Validity
            Not Before: Apr 14 01:33:41 2006 GMT
            Not After : Apr 14 01:33:41 2007 GMT
        Subject: C=JP, O=Osaka University, OU=Cybermedia Center,
CN=Shingo Takeda
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:9f:10:c8:a5:32:36:cc:f7:03:75:5d:6b:24:93:
                    33:eb:04:1a:a6:b0:28:9d:cd:74:3f:06:d2:67:60:
                    4a:13:5c:24:60:25:1a:32:1d:6e:91:e9:47:30:be:
                    aa:96:3a:45:95:81:36:1f:bc:1e:38:4b:e8:d4:ef:
                    d7:df:29:40:83:58:86:8f:e4:9a:2d:9a:6c:ff:b3:
                    5e:9d:5a:ff:b8:47:2c:9d:1e:b2:a9:22:11:40:59:
                    61:bb:bd:9d:23:54:7d:e7:5c:f5:3b:41:eb:55:c2:
                    29:19:fe:13:a1:1f:93:f0:5b:58:08:54:bb:b6:c5:
                    13:a6:b2:1f:ea:d1:9c:f7:9f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            Netscape Cert Type:
                SSL Client, SSL Server, S/MIME, Object Signing
    Signature Algorithm: md5WithRSAEncryption
        42:76:7f:fe:3b:d0:5e:a4:2b:e5:80:80:d2:fa:1d:41:06:ec:
        3a:dc:a7:cc:6d:96:16:6f:77:6e:ce:ad:28:81:7e:cd:40:f2:
        d7:88:00:bd:fc:7c:76:6c:f1:2f:d3:be:b3:b2:2d:b3:75:ea:
        6e:27:a3:83:d7:9e:35:f0:3f:68:56:33:b1:d5:fb:1a:ec:ba:
        01:d4:d4:18:38:7f:16:d4:2a:dd:3d:3c:8c:f0:e3:34:a3:5a:
        48:45:18:03:4f:7b:81:31:57:4d:5c:71:e4:d6:88:bf:8e:14:
        d0:10:50:cb:55:92:4e:dd:ac:74:f2:a0:de:cc:38:35:af:44:
        06:d4
```

図 2.2 X.509 形式のユーザ証明書の例.

グリッドのリソースは CA から発行されたホスト証明書を持ち，サービスはこれを用いてユーザとの相互認証を行う．リソース上の全てのサービスが同一のホスト証明書を使用するとは限らず，それぞれのサービスが異なるホスト証明書を持つこともある．図 2.3 にホスト証明書の例を示す．これは Biogrid CA から発行されたサービス host/scorpio.ais.cmc.osaka-u.ac.jp のホスト証明書である．このサービスの識別名は

```
/C=JP/O=Osaka University/OU=Cybermedia Center
/CN=host/scorpio.ais.cmc.osaka-u.ac.jp
```

と表される．この証明書の有効期限は 1 年である．ホスト証明書にはサービスの公開鍵が含まれており，認証や暗号通信路の確立に使用される．これに対応する秘密鍵はサービスを提供するコンピュータ上に保存されているが，ユーザ証明書とは異なり暗号化はされていない．そのため，コンピュータの管理者（root）にのみアクセスを許可するようにオペ

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3 (0x3)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=JP, O=Osaka University, OU=Cybermedia Center,
CN=Biogrid CA
        Validity
            Not Before: Aug 11 06:46:37 2006 GMT
            Not After : Aug 11 06:46:37 2007 GMT
        Subject: C=JP, O=Osaka University, OU=Cybermedia Center,
CN=host/scorpio.ais.cmc.osaka-u.ac.jp
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:a6:08:0b:4b:a6:98:03:a2:11:56:99:00:ee:a6:
                    dd:1a:b5:dc:7c:1e:b9:5c:5f:7e:18:01:88:d5:34:
                    cf:0e:e0:01:bc:f1:7a:62:2f:1e:12:4d:67:db:06:
                    04:d7:4c:a5:24:4f:f7:3c:e6:d2:0c:42:3c:84:b5:
                    16:7b:fd:b3:3d:60:c0:40:9c:1b:65:b8:0f:cd:ea:
                    a0:d3:d6:6c:42:93:99:d0:25:29:e3:53:16:41:86:
                    a6:66:42:56:d3:3a:7f:63:7e:04:92:8e:24:c0:35:
                    91:ae:de:4f:e6:b0:3a:dc:b7:52:74:7f:3e:01:ef:
                    82:99:8e:5f:f5:44:d5:4b:cb
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            Netscape Cert Type:
                SSL Client, SSL Server, S/MIME, Object Signing
    Signature Algorithm: md5WithRSAEncryption
        ba:b0:37:b3:a2:16:ec:2a:39:11:14:1d:08:05:5b:d0:fe:d6:
        3c:e7:83:0c:f5:f1:25:a4:d1:bc:a1:e4:ba:18:a5:d2:4a:57:
        88:3c:a9:3b:4a:29:07:3f:94:71:bd:82:db:b7:68:8e:3f:fd:
        a6:dd:68:d2:26:6c:26:8f:3c:dd:db:bf:15:36:23:fc:ee:0b:
        60:54:43:15:47:af:3a:dd:6c:ab:97:be:25:a4:80:06:19:6c:
        e5:3c:5b:dc:d4:dd:a3:02:c3:de:bb:c5:ca:79:5c:f0:66:3e:
        be:e5:f3:6f:20:6a:f9:00:dd:53:12:7a:cf:c8:80:41:77:42:
        80:e1
```

図 2.3 X.509 形式のホスト証明書の例.

レーティングシステムの機能を使用して制限される.

これらのユーザ，ホスト証明書を用いてグリッドでのシングルサインオンを実現するために，さらにグリッド認証基盤では X.509 証明書を拡張したプロキシ証明書 [18] が標準化されている．ユーザ証明書は 1 組織あるいは複数組織によって運営される CA によって発行されるが，プロキシ証明書はユーザが発行を行う．ユーザがグリッドの利用を始めるときには，ユーザはまず暗号化されたユーザ秘密鍵のパスフレーズを入力し，プロキシ証明書の発行を行う．以後，このプロキシ証明書とサービスのホスト証明書の間で相互認証が行われるため，ユーザは認証操作を意識することなくリソースを利用することができる．プロキシ証明書情報の例を図 2.4 に示す．これは，図 2.2 のユーザが発行したものであり，図中 issuer の項目がユーザ識別名となっている．subject の項目にはユーザ識別名の末尾に一意な数値を追加したものとなっており，これはプロキシ証明書の識別名である．このプロキシ証明書は有効期限 12 時間で生成され，timeleft の項目は有効な残り時

```
subject   : /C=JP/O=Osaka University/OU=Cybermedia Center
/CN=Shingo TAKEDA/CN=242970403
issuer    : /C=JP/O=Osaka University/CN=Shingo TAKEDA
identity  : /C=JP/O=Osaka University/CN=Shingo TAKEDA
type      : Proxy draft compliant impersonation proxy
strength  : 512 bits
path      : /tmp/x509up_u500
timeleft  : 11:58:37
```

図 2.4 プロキシ証明書情報の例.

```
subject   : /C=JP/O=Osaka University/OU=Cybermedia Center
/CN=Shingo TAKEDA/CN=242970403/CN=129583097
issuer    : /C=JP/O=Osaka University/OU=Cybermedia Center
/CN=Shingo TAKEDA/CN=242970403
identity  : /C=JP/O=Osaka University/OU=Cybermedia Center
/CN=Shingo TAKEDA
type      : Proxy draft compliant limited proxy
strength  : 512 bits
path      : /home/shingo/.globus/job/(中略)/x509_up
timeleft  : 11:53:19
```

図 2.5 委譲証明書情報の例.

間を表しており 11 時間 58 分 37 秒である．ユーザ証明書の有効期限は一般的に数か月から数年が設定されるのに対し，プロキシ証明書の有効期限は数時間から数日と短い．これは，リソース上に一時的に保存される暗号化されていない秘密鍵が盗用された場合のリスクを低減するためである．

委譲証明書はプロキシ証明書の一種で，リソース間の認証においてもシングルサインオンを可能にするために，プロキシ証明書から連鎖的に発行される証明書である．図 2.4 に示したプロキシ証明書から発行された，委譲証明書情報の例を図 2.5 に示す．図中 issuer の項目はプロキシ証明書の識別名となっており，subject にはさらに末尾に異なる数値が付加されている．有効期限は初めに生成したプロキシ証明書のものが引き継がれる．委譲証明書からさらに委譲証明書を生成することも可能である．

プロキシ証明書を用いたシングルサインオンの手順を図 2.6 に示す．まず，ユーザはグリッドの利用開始時にユーザ証明書に対応する秘密鍵のパスフレーズを入力してプロキシ証明書を発行し，それをユーザプロキシに登録する（図中 1）．グリッド認証基盤に対応したクライアントのプロセスは，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コンピュータ上のプロセスと認証を行うため，異なるサービス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（図中 2）．プロキシ証明書から連鎖的に委譲証明書を

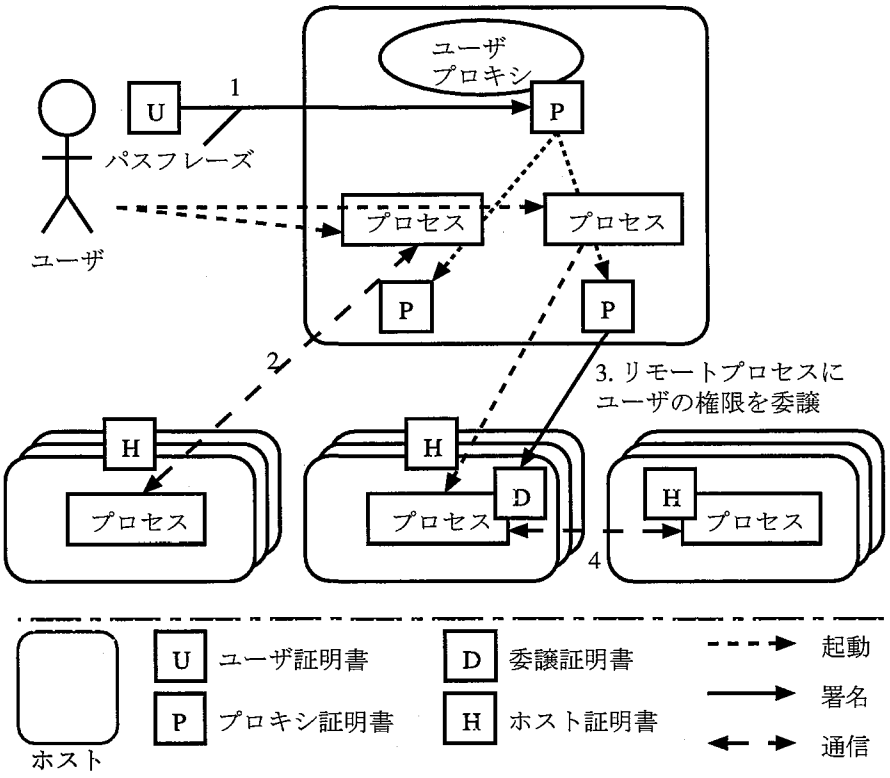

図 2.6 プロキシ証明書を用いたシングルサインオン.

他のコンピュータ上のプロセスに発行することで，ユーザの権限をプロセスに委譲することができ（図中 3），他のコンピュータ間の認証においてもシングルサインオンが有効である（図中 4）.

委譲証明書を含め，プロキシ証明書にはその証明書をどのように利用できるかというポリシを記述するフィールドと，そのポリシがどのような言語で記述されているかを記述するフィールドが用意されている．しかし現時点ではポリシの記述方法の標準化が行われておらず，相互運用性が確保できないため利用されていない.

2.2.3 Grid Security Infrastructure（GSI）

グリッド構築のためのミドルウェアとして，Globus Alliance [19] が開発している Globus Toolkit [20, 21] が学術機関を中心として広く利用されている．Globus Toolkit はグリッド上での分散計算に必要な，セキュリティ，データ転送，リソース管理などの機能を提供する．グリッド技術は相互運用性を高めるために Open Grid Forum（OGF）[22] で標準化が進められており，Globus Toolkit はその標準をいち早く実装していることから，グリッド構築ミドルウェアの事実上の標準技術（デファクトスタンダード）となっている．学

術機関を中心として推進されている研究プロジェクトには，グリッドの基盤技術を研究する Enabling Grids for E-sciencE（EGEE）[23]，TeraGrid [24]，医療への応用を試みる Biomedical Informatics Research Network（BIRN）[25]，気象予測への応用を試みる Earth System Grid（ESG）[26] などがあり，これらをはじめとして今日推進されている数多くの研究開発プロジェクトで Globus Toolkit が採用されている．

Globus Toolkit においてセキュリティ機能を提供するコンポーネントは Grid Security Infrastructure（GSI）[27] である．GSI はグリッド認証基盤の実装であり，グリッドでシングルサインオンを実現するためのライブラリとツールが利用できる．データ転送やリソース管理を実現する Globus Toolkit の他のコンポーネントは，すべてこのライブラリを使用して開発されている．ユーザもこのライブラリを用いて科学技術計算プログラムを開発することで，シングルサインオンで利用できるサービスとしてこのプログラムをグリッド上に展開することができる．ライブラリは C 言語と Java の Application Programming Interface（API）が公開されている．C 言語から API を使用するためには，提供されているヘッダファイルをインクルードし，ライブラリをリンクする．Java から API を使用するためには，classpath に Java Cog Kit [28] へのパスを追加する．これらのライブラリは OpenSSL [29] を使用しており，プログラムを作成するには公開鍵暗号，共通鍵暗号，ネットワーク，ソケット，GSI，OpenSSL に関する知識を要する．管理者によるユーザ証明書とホスト証明書の発行，ユーザによるプロキシ証明書の発行など，証明書に関する操作には GSI のツールを使用する．

GSI はグリッド認証基盤だけでなく，単純な認可機能も提供している．この機能はプロキシ証明書を用いた相互認証に成功した後，ユーザ識別名をオペレーティングシステム上のユーザにマッピングするものである．これによって，ファイルアクセスやネットワーク利用の制御をオペレーティングシステムの機能を使って実現できる．例えば，グリッドのユーザ

```
/C=JP/O=Osaka University/OU=Cybermedia Center/CN=Shingo Takeda
```

がオペレーティングシステム上のユーザ

```
shingo
```

にマッピングされる．これによって，所有者が `shingo` のプロセスとして計算プログラムを起動することで，オペレーティングシステムの機能を使ってファイルやネットワーク利用の制御を行うことができる．マッピングに関する情報は `grid-mapfile` という名前のファイルにリソースの管理者がテキスト形式で記述する．

2.3 シングルサインオンでの並列分散計算

グリッド構築のためには，グリッド認証基盤に加え様々な技術を応用したソフトウェアが組み合わせて用いられる．本節では，グリッドシステム全体としてユーザと管理者の安全性と利便性の両立を困難にしている点を分析するため，グリッド認証基盤上で科学技術計算の並列分散を行うために用いられるソフトウェアの調査を行う．まず，学術機関などで科学技術計算の高速化のために共有されている計算リソースの特性を調査する．そして，それらの計算リソースをグリッド技術で共有する環境について述べる．

2.3.1 共有される計算リソース

現在，グリッドで共有されている計算リソースの多くはクラスタである．クラスタは，汎用的なPCを多数集めてLocal Area Network（LAN）で接続し並列計算を行うシステムで，特殊なプロセッサを持つスーパコンピュータに比べて安価である．また，クラスタを構成する個々のコンピュータ（ノード）はPCであるため，日常的にPCを利用しているユーザにとって扱いやすいという利点もある．40ノードから構成されるクラスタの例を図2.7に示す．この例では，それぞれのノードはプロセッサを2基持った1Uサイズのラックマウント型PCサーバで，80のプロセスを同時に実行することができる．たんぱく質と化合物の結合シミュレーションなど，アプリケーションのパラメータを変えて多数の試行錯誤を行う処理はプロセス間の通信が不要であるため，このようなクラスタにおいてノード数に比例した高速化が可能である．この特性から，クラスタはプロセス間の通信が発生しない独立した多数のプロセスを分散計算する用途に適している．複数組織の計算リソースをグリッドで共有すると，インターネットを使用するために組織間の通信オーバヘッドも大きい．このために，グリッドはプロセス間の通信が必要な計算の分散には不向きであり，今日のグリッドでは複数の組織間でクラスタを共有し，独立性の高い計算を分散処理することが行われている．本研究では，このようなリソースに分散する科学技術計算を行う際の，ユーザと管理者の負担を低減することを考える．

一般的に，図2.8に示すようにクラスタにはクラスタの制御を行う管理ノードが1台設置され，残りは計算のみを行う計算ノードとして使用される．クラスタを複数ユーザで共有する場合，競合を避けるために管理ノードにローカルスケジューラが導入される．ローカルスケジューラは，計算を開始から終了して結果を返すまでをジョブと呼ばれる単位で管理し，定められたポリシに基づいてジョブを計算ノードに割り当てる．ローカルスケジューラ機能を持つ代表的なソフトウェアにPortable Batch System（PBS）[30]，Sun Grid Engine（SGE）[31]，Condor [32]，Platform LSF [33]などがある．例として，ユー

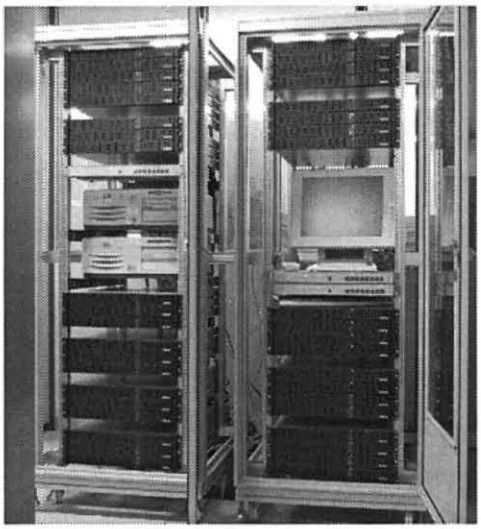

図 2.7 40 ノードから構成されるクラスタの例.

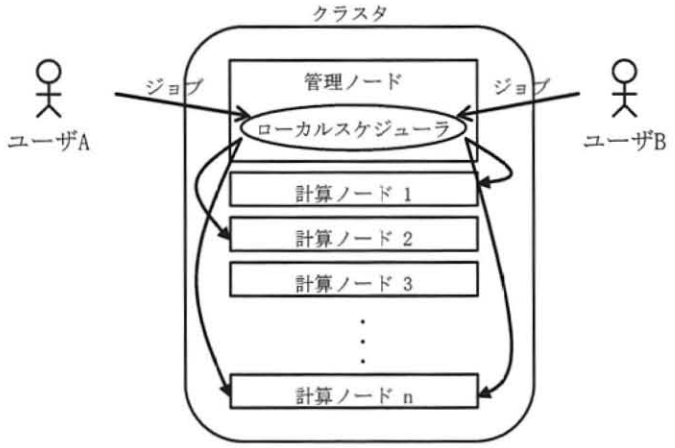

図 2.8 クラスタの構成とローカルスケジューラによる競合の回避.

ザが SGE で並列ジョブを実行するコマンドの例を示す.

```
$ qsub -np 20 myjob.sh
```

ここで，-np オプションはプロセッサ数（Number of Processors）を意味し，CPU をいくつ確保するかを指定している．myjob.sh はシェルスクリプトであるが，コメントとして SGE が解釈できる設定が埋め込まれており，これに基づいて SGE はジョブの配置を行う.

2.3.2 グリッド認証基盤上での計算リソース共有

現在，グリッド構築のためのミドルウェアとして，Globus Toolkit がデファクトスタンダードとして学術機関を中心に広く利用されている．組織外のクラスタでジョブを実行するためには，Globus Toolkit の Grid Resource Allocation and Management（GRAM）[34] が使用される．GRAM は GSI に対応したサービスで，グリッドのユーザからのジョブ実行要求を受け付け，ローカルスケジューラに橋渡しする役割を持つ．例として，ユーザが組織外のクラスタで並列ジョブを実行するコマンドの例を示す．

```
$ globusrun remote.gramserver.net/jobmanager-sge \
    "&(executable=/home/me/myjob.sh)(count=20)"
```

ダブルクォートで囲まれた部分は Resource Specification Language（RSL）[35] と呼ばれる言語で，どのようにジョブを実行するかが記述されている．このようなコマンドを使用してのジョブの実行要求の送信は利便性に欠けており，利便性を向上させるためにグリッドポータルとメタスケジューラが設置されることが一般的である．

グリッドポータルはユーザにグリッドへのインタフェースを提供する Web アプリケーションである．図 2.9 に GridSphere [36] で構築されたグリッドポータルの例を示す．GridSphere はグリッドポータル構築のための代表的なフレームワークで，ポータル開発者はアプリケーションを Portlet として開発することで機能を追加することができる．グリッドポータルは複雑なコマンドを隠蔽し，ユーザの視覚的な操作を可能とする．また，Web ブラウザのみで利用でき，ユーザのクライアントに追加のソフトウェアを導入する必要がないという利点もある．図のログイン画面ではユーザ ID とパスワードを入力するようになっており，グリッドポータルではこの形態が一般的である．そのため，プロキシ証明書によるシングルサインオンを実現するために MyProxy [37] などの証明書リポジトリが使用される．ユーザはあらかじめプロキシ証明書をリポジトリに預けておき，グリッドポータルがリポジトリからプロキシ証明書を取得することでシングルサインオンが利用可能となる．

メタスケジューラは，定められたポリシに従って自動的にリソースを選択するサービスである．ユーザが自身でクラスタの利用状況などを確認し，ジョブの実行要求を送信するクラスタを選択する必要があると利便性に欠ける．そこで，管理者は図 2.10 に示すようにメタスケジューラを設置し，ユーザがメタスケジューラにジョブ実行要求を送信するだけで，自動的に最適なクラスタに要求が転送されるようにする．Condor-G [38] はこのメタスケジューラ機能を備えた代表的なソフトウェアの一例であり，SGE や Platform LSF もローカルスケジューラ機能に加えてメタスケジューラの機能も有している．

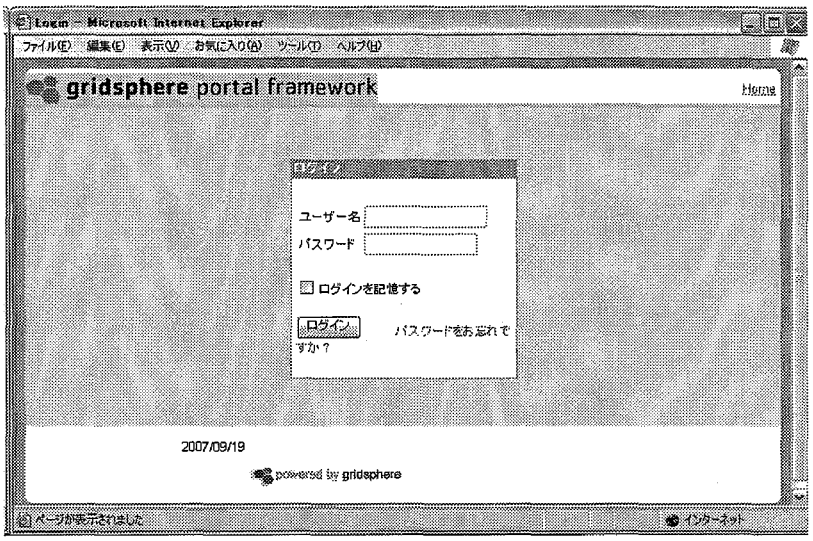

図 2.9 GridSphere で構築されたグリッドポータルのログイン画面の例.

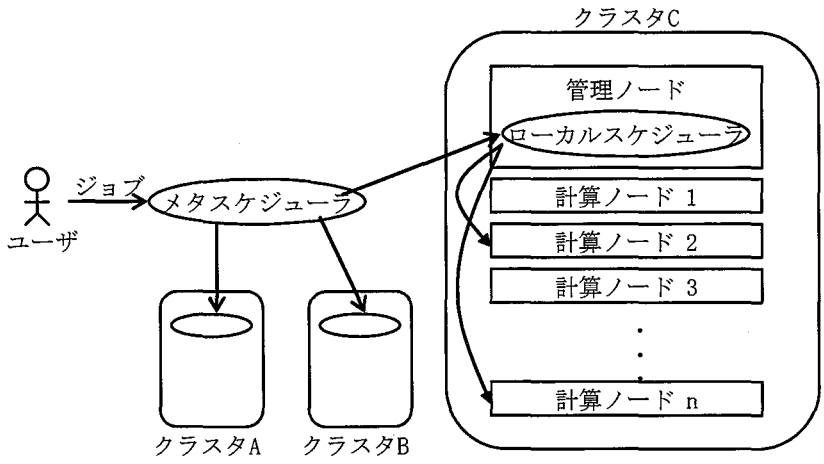

図 2.10 メタスケジューラによる計算リソースの選択.

このように構築された分散計算のためのグリッドでは，管理者らが各クラスタの稼働状況を把握し，適切にスケジューラのポリシを調節する必要がある．これを支援するために，多くのグリッドではリソースモニタが導入されている．リソースモニタは CPU やメモリの使用率，コンピュータの稼働率などを可視化し，リソースの状態をユーザと管理者に示すものである．グリッドを対象とする代表的なリソースモニタに SCMSWeb [39] がある．SCMSWeb はクラスタを構成するコンピュータからリソース情報を階層的に収集し，グリッドのような大規模な環境でも一元的にリソース情報を Web インタフェース上に可視化することができる．SCMSWeb の可視化画面の例を図 2.11 に示す．この例では，ApGrid，PRAGMA，ThaiGrid の 3 つの研究プロジェクトで構築されたグリッドの情報を

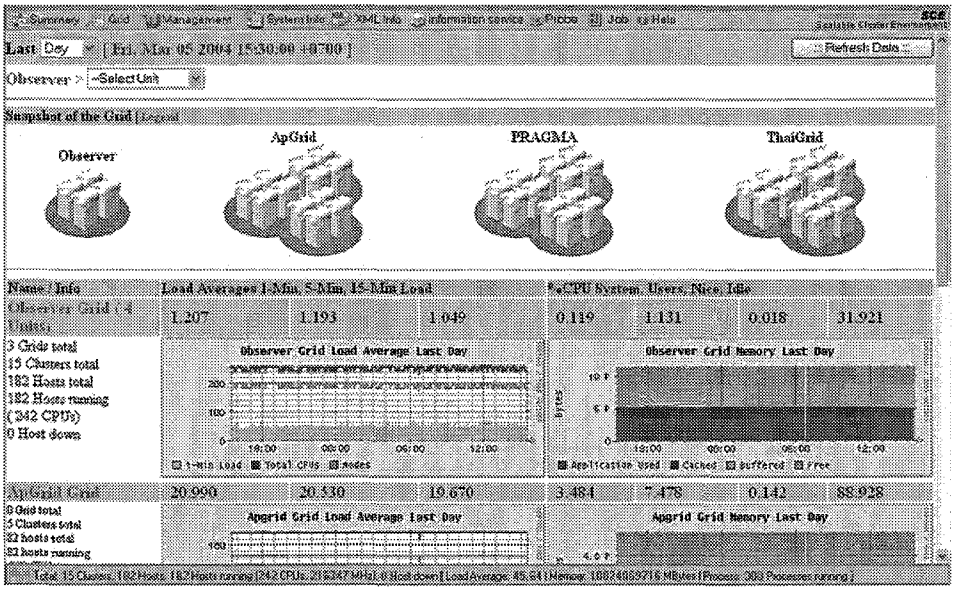

図 2.11 SCMSWeb の可視化画面．文献 [39] より引用．

まとめて表示しており，全体の CPU とメモリの使用率を知ることができる．

2.4 グリッド構築上の問題と課題

これまでに述べたように，グリッド構築ではグリッド認証基盤の上に，GRAM，メタスケジューラ，グリッドポータルなどが導入され，これらを利用することで複数組織のクラスタ上での並列分散計算が可能となる．しかし，1 章で述べたように，安全性の確保にかかる負担の重さがグリッド利用の障壁となっており，ユーザと管理者の両視点からグリッド利用の負担を軽減する技術が求められている．本節では，2.2 節と 2.3 節で説明したグリッド環境におけるユーザと管理者のそれぞれの問題を分析し，解決のための課題を導く．

2.4.1 ユーザ視点からの安全性と利便性

グリッドポータルやスケジューラは，複雑なコマンドや計算が実行されるリソースの位置をユーザから隠蔽し，ユーザがこれらを意識することなく利用することを可能にする．しかし，ユーザが計算プログラムを記述し，独自の計算を行う場合は，プログラムと計算に必要なデータをユーザが配置しなければならない．オペレーティングシステム上ではプログラムはファイルとして存在し，また多くの科学技術計算プログラムはデータ入出力にファイルを用いる．プログラムとデータの配置において，プログラムファイルの転送では整合性の確保，データファイルの転送では機密保護が重要となる．プログラムは計算リ

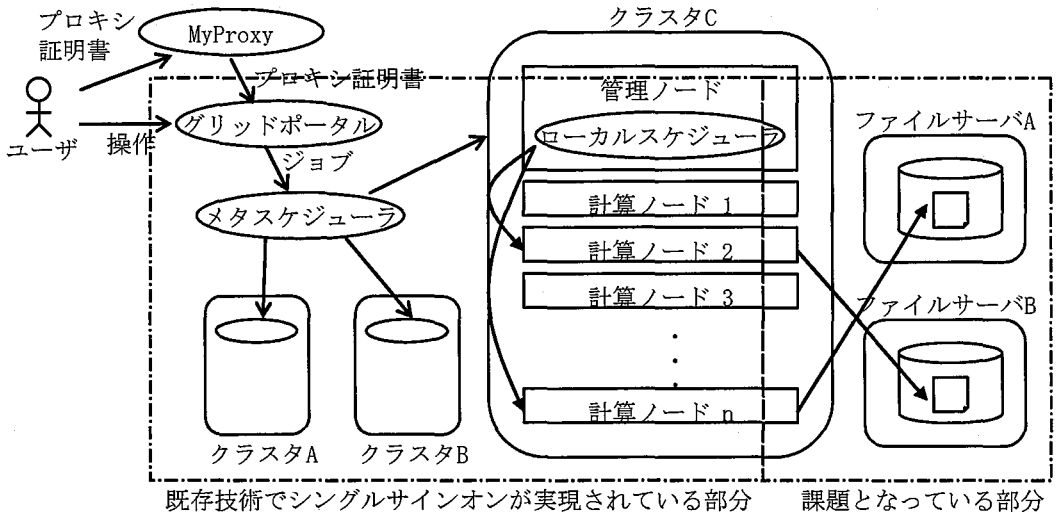

図 2.12 シングルサインオンの科学技術計算に向けての課題.

ソース上で実行されるため，通信経路で改ざんされると，不正利用の被害が発生する可能性がある．また，1 章で記したように，グリッド技術の医療や産学連携プロジェクトへの応用も広がっており，データの機密保護が必要な計算も増加している．このため，安全性に配慮しながらスケジューラによって選択しうるすべての計算リソースにプログラムとデータを配置することには，多大な時間と労力を要する．

また，ユーザは，これまでに Fortran や C 言語で記述したプログラムをそのままグリッドで利用したいと考えるが，拡張が必要な場合がある．ユーザの要求するジョブの中には，図 2.12 に示すような計算リソースとは別のファイルサーバにあるデータを必要とするものがあり，このようなジョブ特性をもつ既存プログラムに対してはシングルサインオンによる利便性は確保できない．例えば，計算リソースとは別のリソース上にあるファイルを入力とし，計算リソース上のファイルに出力する処理において，ユーザが計算を要求してから結果が得られるまでの一連の過程をシングルサインオンで実現するための方法として，以下の 3 つが考えられる．

1. Globus Toolkit の API を使用した計算プログラムの拡張
2. Globus Toolkit のコマンドを使用した入力ファイルの計算前転送
3. 遠隔ファイルシステムの静的マウント

以下ではそれぞれの方法について考察する．

計算プログラムを GSI に対応させるには，Globus Toolkit のライブラリを使用して計算プログラムを拡張する．図 2.13 に計算プログラムを Globus Toolkit に対応させ，外部ファイルを処理する例を示す．この方法では，クラスタ A の計算プログラムが Globus

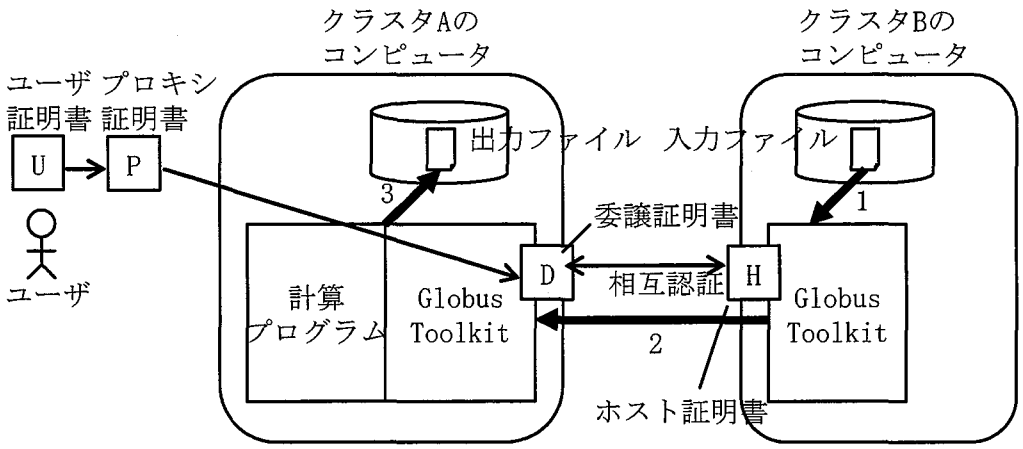

図 2.13 Globus Toolkit の API を使用した計算プログラムの拡張.

Toolkit の API を呼び出し，クラスタ B のファイル転送サービスにアクセスしている．クラスタ B の Globus Toolkit は入力ファイルを読み取り（図中 1），ネットワークを介してクラスタ A の Globus Toolkit にデータを転送する（図中 2）．計算プログラムはこのデータを処理し，結果をファイルに出力する（図中 3）．クラスタ A にはユーザの委譲証明書があり，これを使用してファイル転送サービスでの認証が行われるため，シングルサインオンが実現される．この方法では，計算プログラムの開発に暗号技術やネットワークプログラミングに関する詳しい知識が求められ，情報科学を専門としないユーザにとっては負担が大きいという問題がある．

計算前にファイルを転送しておく方法では，シェルスクリプトを用いて比較的簡単にシングルサインオンを実現できる．図 2.14 にシェルスクリプトで Globus Toolkit を用いて外部ファイルを処理する例を示す．この方法では，まず Globus Toolkit のファイル転送コマンドと計算プログラムを起動するシェルスクリプトを記述しておき，このシェルスクリプトのプロセスに権限を委譲する．ファイル転送コマンドはシェルスクリプトの子プロセスとして起動されるため，クラスタ B のコンピュータ上の Globus Toolkit のファイル転送サービスと委譲証明書を使用してシングルサインオンで認証が可能である．クラスタ B のファイルは Globus Toolkit に読み取られ（図中 1），ネットワークを介してクラスタ A の Globus Toolkit に転送される（図中 2）．このファイルは一時的にクラスタ A に格納され（図中 3），それを計算プログラムが読み取る（図中 4）．この方法は前者の方法に比べて実現は容易であるが，転送が完了するまで計算を開始できず，一時保存するためのストレージが必要で非効率であるという問題がある．

データリソースのファイルシステムを静的にマウントする方法では Globus Toolkit は使用しない．オペレーティングシステムの機能を使用してファイルシステムをマウントすれ

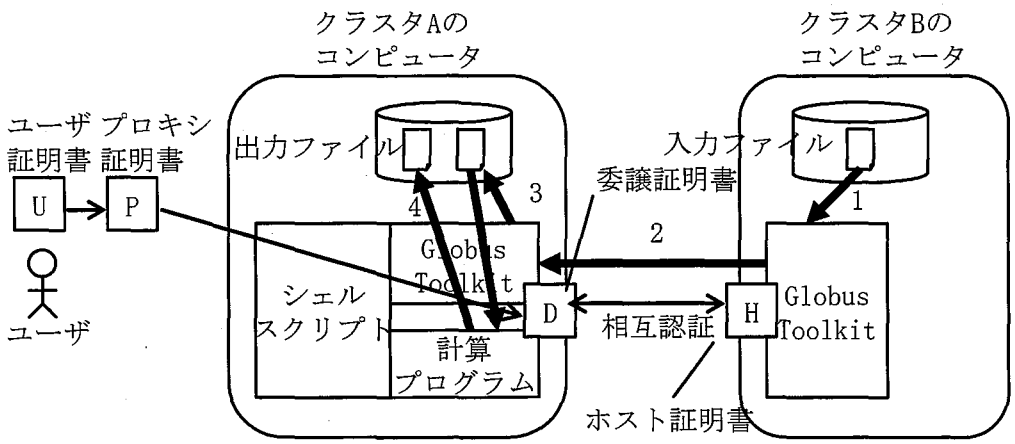

図 2.14 Globus Toolkit のコマンドを使用した入力ファイルの計算前転送.

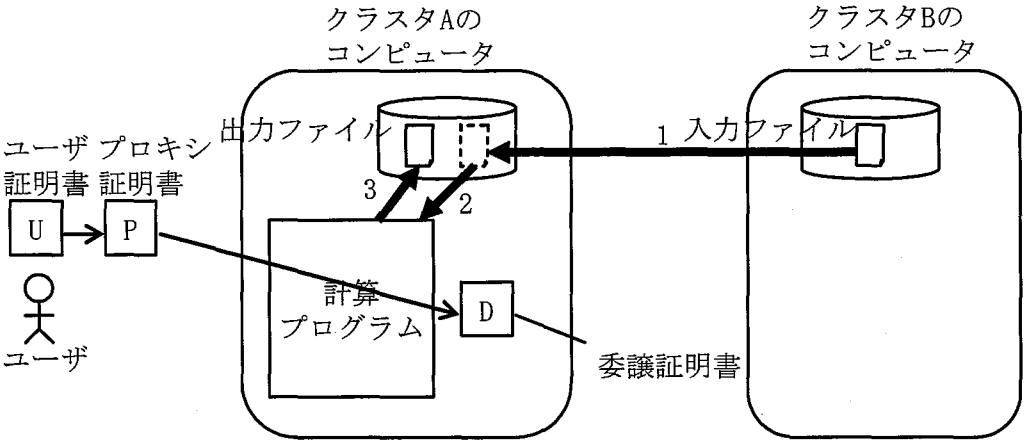

図 2.15 遠隔ファイルシステムの静的マウント.

ば，計算プログラムからはあたかもアクセスするファイルが同じファイルシステム上にあるかのように扱えるからである．この方法では，計算プログラムをそのまま利用でき，ランダムアクセスや部分的な書き換えの効率にも優れる．しかし，他組織のファイルシステムを静的にマウントすることは現実的ではない．オペレーティングシステムのマウント機能は管理者権限で設定する必要がありユーザが自由に利用できず，他組織にマウントを許可することは安全性の低下を招くためである．

シングルサインオンで計算を行う際，これらの 3 つのいずれの方法でも，ユーザ利便性とシステムの安全性を両立することができない．グリッド認証基盤や，グリッドポータル，スケジューラの整備に伴ってユーザによるグリッド利用の利便性は向上しているが，ユーザが独自の計算プログラムとデータを展開して分散計算を行うときには，ユーザはファイルの位置を意識する必要があり，このことが利便性低下の要因となっている．ユー

ザ利便性と安全性の両立のためには，グリッド認証基盤に適応し，安全性を犠牲にすることなくファイルの位置を意識させない，つまり，位置透過的なファイルアクセス手法の実現が最大の課題となっている．

2.4.2 管理者視点からの安全性と利便性

ユーザがグリッドポータルから計算の実行を要求すると，メタスケジューラは複数の組織から適切なクラスタを選択し，さらに各クラスタのローカルスケジューラはジョブを複数のノードに割り当てて並列計算が行われる．それぞれのノードで動作する計算プログラムは，委譲証明書を用いて，さらに別のリソースを要求することもある．このとき，委譲証明書ではシングルサインオンを実現するために秘密鍵が暗号化されておらず，委譲証明書が漏えいすると正規ユーザの権限でグリッドのリソースを不正利用される可能性がある．図 2.16 に委譲証明書の盗用による不正利用の例を示す．ここでは，図中のユーザ A がグリッドポータルとメタスケジューラを経由して計算リソース B に計算を要求している．計算プログラムは他のリソースに保存されているデータを利用するため，シングルサインオンを実現するために計算リソース上に委譲証明書が置かれる．この委譲証明書はオペレーティングシステムの機能を使用して，同一リソース上の他のユーザからはアクセスできないように設定される．しかし，設定の誤りやソフトウェアの不具合が存在すると，グリッドを利用する他のユーザ，または外部から侵入したユーザが他ユーザの委譲証明書にアクセスできる場合がある．実際，Globus Toolkit の過去のバージョンには脆弱性があり，シンボリックリンクを作成することで他ユーザの証明書にアクセスできる不具合が存在した [40]．このような方法で委譲証明書が盗用されると，不正利用の被害が発生する可能性がある．

一般的に，このような不正利用の被害を小さくするためにはプログラムに必要最小限の権限を与えることが原則であり，このことは最小特権の原理（Principle of least privilege）と呼ばれている [41]．本研究で対象とする不正利用では，委譲証明書を持つプログラムにユーザの要求する計算に必要な最小限の権限のみを持たせることが理想である．しかし，証明書のポリシ記述方法を標準化する作業は OGF で行われているものの，現在は標準となるものが無く，独自の方法で記述すると相互運用性が失われる．このため，委譲証明書の権限は有効期間のみで制限されているのが実情であるが，メタスケジューラからローカルスケジューラへと多段にジョブの振り分けが行われ，それぞれ計算能力の異なるリソースで計算が行われるグリッドでは，計算に必要な時間を正確に推定することが困難である．

これらの理由から委譲証明書の権限を厳密に制御することは困難であるため，グリッドの安全な運用のためには管理者がグリッドの状態をモニタリングし，不正利用や機密漏え

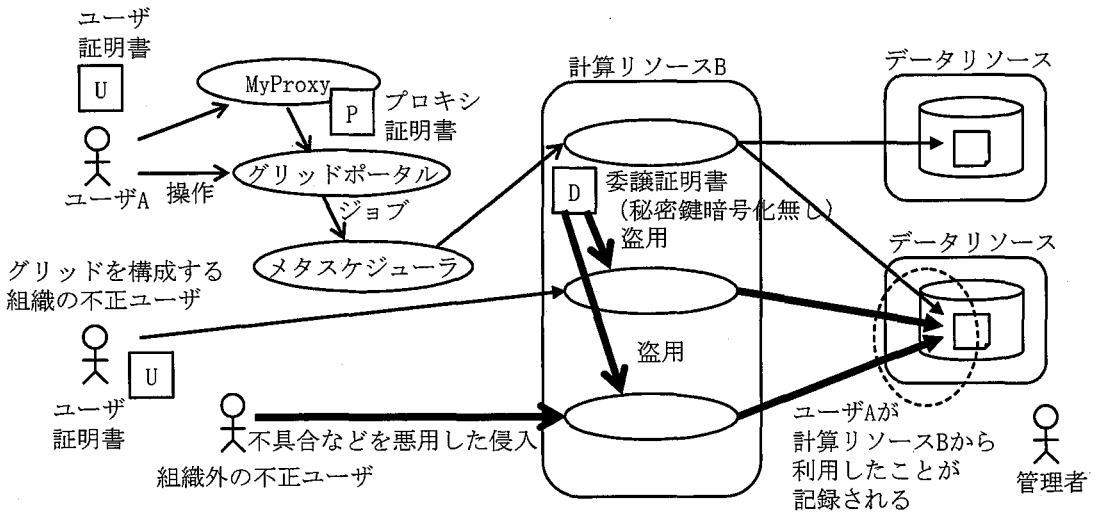

図 2.16 委譲証明書の盗用による不正利用の例.

いを事前に防止する作業が重要となっている．リソース状態のモニタリングのためには，SCMSWeb などのリソースモニタが利用される．これらは，管理者が容易に CPU やメモリの使用量を把握し，スケジューラのポリシを調整することなどに利用できる．しかし，リソースモニタは認証や認可の情報を扱わないため，不正利用や機密漏えいに関する事象を発見するためには，管理者がグリッド認証基盤から各リソース上に出力されるログファイルを調査する必要がある．

ログファイルには，時刻，リソースを要求したユーザ，要求送信元のクライアント，認証と認可の成否，要求したジョブの内容などが要求ごと記録される．大規模な計算では，メタスケジューラとローカルスケジューラによってユーザの計算要求が多数のジョブに細分化され，複数の計算リソースに割り当てられるため，ログファイルには多くの出力がなされる．このために，管理者が調査するログファイルは大きなものとなる．また，単一のリソースに出力されるログ情報からは正当な利用と不正利用の区別が付かない場合がある．図 2.16 に示した例では，不正利用においてデータリソースにはユーザ A が計算リソース B からリソースを要求したことが記録されるが，これは正当な利用で出力される情報と同じである．このような不正利用を発見するためには，管理者が複数リソースのログ情報から統合的に判断することが求められる．日常的に複数リソースに分散したログファイルを収集し，このような分析を行って不正利用を調査することには多大な労力が伴い，管理者による安全なグリッド運用の妨げとなっている．

以上の問題点をまとめると，管理者にとって利便性の高いグリッドを安全に運用する上での問題は，セキュリティ状態の把握と，リソース要求の分析が困難であることである．このことは，グリッド認証基盤をセキュリティの観点からモニタリングし，管理者の状態

把握，及び，分析を支援するセキュリティモニタリング技術の実現は急務であることを意味している．

2.5 むすび

本章では，グリッドで科学技術計算を並列分散できる安全かつ利便性の高いグリッドを構築する上で，ユーザと管理者にとって利便性を低下させる原因となっている点を分析し，改善するための課題を導出した．グリッドでは，プロキシ証明書による認証機能を備えたグリッド認証基盤の上で各種サービスが提供され，ユーザはシングルサインオンで複数組織に分散したリソースを利用することができる．グリッド認証基盤の実装としては，GSI がデファクトスタンダードとして学術機関を中心に広く用いられている．

グリッドで共有される計算リソースの多くはクラスタであり，クラスタは独立した多数のジョブの並列処理に適している．クラスタを複数ユーザで共有する際には，競合を避けるためにローカルスケジューラが使用され，ローカルスケジューラは定められたポリシに従って自動的にジョブをノードに振り分ける．シングルサインオン分散計算環境の構築のためには，グリッド認証基盤だけでなく，利便性を向上させるために様々なソフトウェアが組み合わせて導入される．グリッドポータルはユーザに直感的なインタフェースを提供し，メタスケジューラは自動的に最適な計算リソースを選択する．リソースモニタは管理者がリソース利用状態を把握することを容易にする．

このように構築されたグリッドで，ユーザがシングルサインオンの分散計算を実現する上での最大の問題は，ファイルの位置と安全性を意識する必要があることである．スケジューラは自動的に最適な計算リソースを選択するが，これは計算プログラムと計算に必要なデータがあらかじめ計算リソースに配置されていることが前提である．そのため，ユーザが自身で記述した独自の計算プログラムを実行する場合には，スケジューラによって選択される可能性のあるすべてのリソースにプログラムとデータを安全に配置しておかなければならない．また，プログラムが動作しているリソースとは別のリソース上にあるデータが必要な計算では，GSI のライブラリを用いてプログラムを拡張するなどの作業が発生し，ユーザの大きな負担となる．そこで，グリッド認証基盤に適応し，高い安全性と位置透過性を兼ね備えたファイルアクセス技術の実現が課題となっている．3 章では，この課題を解決するファイルアクセス手法の提案を行う．

一方，管理者がグリッドを安全に運用する上での最大の問題は，シングルサインオンによってセキュリティ状態の把握と分析が困難となっていることである．グリッドでユーザが計算を要求すると，要求は細分化，分散され，複雑かつ多数のリソース要求が発生し，それぞれにおいて認証，認可が行われる．リソースモニタはグリッド認証基盤から情報を取得しないためにセキュリティ状態を把握するためには適しておらず，管理者はグリッド

認証基盤のログファイルを日常的に分析しなくてはならない．また，証明書の盗用などの不正利用は単独組織の情報だけでは判断することが困難である．そこで，グリッド認証基盤をセキュリティの観点からモニタリングし，管理者の状態把握，及び，分析を支援するセキュリティモニタリング技術の実現が課題となっている．4章では，この問題を解決するモニタリングシステムの提案を行う．

第3章

グリッド認証基盤に適応した分散ファイルシステム

3.1 まえがき

グリッドは医療や産学連携プロジェクトにも応用が推進されており，個人情報や企業秘密の保護が課題となっている．一方，学術機関では科学技術計算を短時間で行うために高性能コンピュータが高速ネットワークで接続されるため，このようなコンピュータやネットワークが不正利用され，大きな被害が発生することを防止することも重要な課題である．しかし，グリッドでの科学技術計算では安全性とグリッド認証基盤によるシングルサインオンの利便性の両立が難しいという問題がある．各科学技術分野の専門家らはすでに数多くの計算プログラムを作成しているが，これらをグリッド認証基盤に適応させるためにはグリッドに関する専門的な知識と高いプログラミング能力が要求される．また，計算プログラムと計算に必要なデータを多数のリソースに配置することにも多大な労力と時間を要する．

このような背景から，ユーザ利便性と安全性の両立のために，グリッド認証基盤に適応し，安全性を犠牲にすることなくファイルの位置を意識させない，つまり，位置透過的なファイルアクセス手法の実現が課題となっている．本章では，この課題を解決し，グリッド技術を用いた学際研究を容易にすることを目的に，安全性と位置透過性を両立したファイルアクセス手法を提案する．

以下に本章の構成を示す．3.2 節では，グリッドでの並列分散計算を，ユーザがファイルの位置を意識せず安全に行うためのファイルアクセス手法に対するユーザの要求を分析する．3.3 節では，3.2 節の要求分析の結果に基づき，グリッドで利用されている既存ファイルアクセス手法と，ファイル共有に利用されている分散ファイルシステムの安全性と位置透過性を調査する．3.4 節では，既存の分散ファイルシステムをグリッド認証基盤に適

応させる手法を提案し，その設計と実装について説明する．3.5 節では，実装したシステムを，安全性，位置透過性，及び，スループットの観点から評価し，提案手法の有用性を議論する．3.6 節では，応用事例として，実装したシステムを用いて生物データを高速に分析するグリッドについて述べる．

3.2 要求分析

本節では，グリッドでの並列分散計算を，ユーザがファイルの位置を意識せず安全に行うためのファイルアクセス手法に対するユーザの要求を分析する．ここでは，位置透過性と安全性のそれぞれの観点から満たすべき要求を分析する．

3.2.1 位置透過性

データファイルへの位置透過的なアクセスを実現するためには，計算プログラムが同じリソース上にあるデータファイルにアクセスする際に用いられる手段と同じ手段で，他のリソース上のデータファイルにアクセスできることが望ましい．同じリソース上にあるファイルへのアクセス手段は，Portable Operating System Interface for UNIX（POSIX）で標準化されており，計算プログラムは POSIX の Application Programming Interface（API）を用いて作成される．すなわち，位置透過的なファイルアクセスを実現するためには，計算プログラムが POSIX の API で他のリソース上のファイルにアクセスできることが望ましい．一方，プログラムファイルへのアクセスでは，プログラムを実行するリソース上にプログラムファイルを一時的にコピーする必要があると位置透過性に欠ける．このため，位置透過的なアクセスを実現するためには，オペレーティングシステムが他のリソース上のプログラムを直接ロードして実行できることが望ましい．

グリッドにおける分散計算では，スケジューラによって計算プログラムを実行する計算リソースが動的に割り当てられる．このため，位置透過性なアクセスのためには，どの計算リソースからも同じようにファイルを指定できなければならない．例えば，WWW で使用されている URL はインターネット上のどのコンピュータからアクセスしても同じファイルに到達できるファイル指定方法の一つである．このように一意なファイル名空間を用い，スケジューラによってジョブがどの計算リソースに割り当てられても，計算プログラムが同じファイルにアクセスできる必要がある．

以上をまとめると，安全性の確保のためには以下の 3 つの要求を満たす必要がある．

要求（r1）POSIX API での遠隔ファイルアクセス

計算プログラムが，同じリソース上にあるファイルと同じ手段で，他のリソース上にあるデータファイルにアクセスできることが望ましい．

要求（r2）オペレーティングシステムによる遠隔プログラムファイルのロード
オペレーティングシステムが他のリソース上のプログラムをロードして実行できる必要がある．

要求（r3）一意なファイル名空間
ファイル名を一意にし，どのリソース上からアクセスしても同じファイル名で同じファイルに到達できる必要がある．

3.2.2 安全性

ユーザは科学技術計算を高速に行うために，複数組織の計算リソースに計算要求を分散させて分散処理を行う．ここで，計算リソース上で動作している計算プログラムのプロセスが，データリソース上のファイルにアクセスするごとにユーザにパスワード入力などの認証操作を求めることは非現実的である．そこで認証では，グリッド認証基盤に適応し，シングルサインオンを実現する必要がある．具体的には，計算プログラムのプロセスが持つプロキシ証明書を取得し，これを用いてデータリソース上のホスト証明書との間で相互認証を行う．

しかし，プロキシ証明書の秘密鍵は暗号化されていない状態でリソース上に存在し，万一これが第三者に漏えいした場合，プロキシ証明書の有効期間内で不正利用の被害を受ける可能性がある．一般的に，プロキシ証明書の有効期限は，ユーザ証明書と比較して短く設定される．計算リソースの不正利用ではこれにより被害を軽減することが可能であるが，データリソースの不正利用では機密データを一度不正に複製されるとその後の複製を阻止することは不可能である．安全性を確保するためには，プロキシ証明書を無条件に受理するのではなく，計算リソースへの権限委譲によって発行された委譲証明書に関してはユーザが許可したもののみ受理できるようにする認可機能が必要である．

認証と認可の後には，送受信されるデータの機密性と整合性の確保が重要となる．グリッド技術は医療や産学連携プロジェクトなど機密データを扱う分野でも応用が推進されており，盗聴による機密漏えいの防止に努めなければならない．また，分散計算では実行ファイルや設定ファイルが転送されるため，改ざんによる不正利用を防止しなければならない．これらの理由から，通信データの暗号化が必要不可欠となる．

以上をまとめると，安全性の確保のためには以下の 3 つの要求を満たす必要がある．

要求（r4）グリッド認証基盤への適応
プロキシ証明書によるシングルサインオンに対応し，公開鍵暗号を用いた安全な認証で利用できる必要がある．

要求（r5）委譲証明書の制限
権限委譲によって生成されたプロキシ証明書である委譲証明書は，ユーザが許可したもの

のみ受理できるようにする必要がある．

要求（r6）機密性と整合性の確保

通信データを暗号化し，盗聴と改ざんを防止する必要がある．

3.3 既存の遠隔ファイルアクセス手法

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，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いる．ここでは，既存のグリッド技術と分散ファイルシステムを調査し，3.2 節の要求分析の結果を基に安全性と位置透過性の観点で評価する．表 3.1 は本章における比較をまとめたものである．

3.3.1 グリッドを対象とした既存ファイルアクセス手法

GridFTP [42, 43] は，従来の FTP をデータ転送効率と安全性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” [44] 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．ユーザとサービスの認証には GSI が使用される．

Globus Toolkit 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の高さから，Globus Toolkit で実装されている GridFTP はデータのレプリケーション [45] など，多数のデータ管理サービスで採用されている．Globus Toolkit 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効率が悪い．したがって，効率的なアクセスのためには計算プログラムから API を呼び出すことが求められる．現時点では，Globus Toolkit の GridFTP は RFC 2228 を全て実装してはおらず，データチャネルの暗号化と検証は行わない．

Parrot [46] はネットワーク上のファイルに POSIX の 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）[47]，Castor Remote File I/O（RFIO）[48] 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計算プログラムを起動すると，Parrot は Linux の ptrace デバッギングインタフェースを通じて計算プログラム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 でしか使用できず，オペレーティングシステムが Parrot を使用して他のリソース上のプログラムをロードする

表 3.1 既存手法の安全性と位置透過性の比較.

	ファイルアクセス		データグリッド		分散ファイルシステム		
	GridFTP (Globus)	Parrot (GridFTP)	SRB	Gfarm	NFS	AFS, Coda	SFS
位置透過性							
(r1) POSIX I/O	×	○	○	○	○	○	○
(r2) プログラム実行	×	×	×	×	○	○	○
(r3) 一意ファイル名	○	○	○	○	×	△	○
安全性							
(r4) GSI 認証	○	○	○	○	×	×	×
(r5) 委譲証明書制限	×	×	×	×	×	×	×
(r6) 通信の暗号化	×	×	○	○	×	○	○

こともできない．Parrot の GridFTP は Globus Toolkit の実装を使用しており，GSI の認証で利用できるがデータチャネルの暗号化は行わない．

3.3.2 データグリッド技術

Storage Resource Broker（SRB）[49] 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，高機能なため多くのプロジェクトで採用されている．SRB はデータ操作のための API を提供するほか，グラフィカルユーザインタフェースや Web インタフェースでも操作でき，グリッドに関する専門知識を有さないユーザでも利用できるよう配慮されている．SRB は POSIX API でのアクセスのための動的リンクライブラリも提供しており，これを利用することでプログラムは SRB 上のファイルに POSIX API でアクセスできる．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計算プログラム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計算プログラムでは利用できず，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．オペレーティングシステムが動的リンクライブラリを使用して他のリソース上のプログラムをロードすることもできない．SRB は GSI の認証で利用でき，GSI の機能を使用して通信の暗号化を行える．

Gfarm は Grid Datafarm [50] 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 は高解像度天体望遠鏡から出力される膨大な観測データ [6] 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 はファイル操作のためのコマンドラインツールと，ファイルアクセスのための API を提供している．Gfarm では，Gfarm 上のファイルへ POSIX API でのアクセスを可能とするために，計算プログラム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計算プログラム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．オペレーティングシステムがこの機能を使用して他のリソース上のプログラムをロードすることは原理上できない．Gfarm も SRB と同様に GSI の認証で利用でき，GSI の機能を使用して通信の暗号化と検証を行える．

3.3.3 分散ファイルシステム

ネットワーク上のファイルへの位置透過的なアクセスを実現する方法の 1 つとして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に依存した機能を使用することが多いため，多種のオペレーティングシステム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 が採用している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計算プログラムが多く，オペレーティングシステムが他のコンピュータ上のプログラムをロードすることも可能である．

現在，最も多くのオペレーティングシステムが実装している分散ファイルシステムは Network File System（NFS）である．NFS の認証はコンピュータとコンピュータの間で行われ，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）[51] と，AFS から派生しモバイル向けの改良が行われた Coda [52] 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のみである．NFS ではマウント先のディレクトリは定められていないのに対して，AFS では `/afs`，Coda では `/coda` に全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コンピュータにログインしても同じディレクトリ構造が利用でき，一意なファイル名空間が実現されている．

Self-certifying File System（SFS）[53-55] 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 Remote Procedure Call（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 系オペレーティングシステム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 サーバのコンピュータ名，*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* はサーバの成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一意なファイル名空間を実現している．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，他のユーザによってマウントされたディレクトリを見ることはできず機密性が高い．

3.3.4 要件実現に向けた考察

以上の調査結果によると，グリッドを対象として開発されたファイルアクセス手法に関しては，SRB と Gfarm が 3.2 節で挙げた 6 つの要求のうち，要求（r1），（r3），（r4），（r6）の 4 つを満たしており，ユーザが要求するファイルアクセス手法に最も近い．これらの手法を要求（r2），（r5）を満たすように拡張し，全ての要件を満たすことが考えられるが，いずれの手法の拡張でも原理的に要求（r2）を満たすことは困難である．要求（r2）では，オペレーティングシステムが他のリソース上のプログラムをロードして実行できることが求められている．このためには，ファイルアクセスを提供するソフトウェアはオペレーティングシステムのカーネルモードで動作する必要があるが，グリッドを対象としたファイルアクセス手法は全てオペレーティングシステムのユーザモードで動作するように実装されている．グリッドでは多種のオペレーティングシステムが混在して利用されているため，カーネルモードで動作するソフトウェアは互換性が問題となり，汎用性の確保が困難となるためである．

分散ファイルシステムでは，SFS が要求（r1），（r2），（r3），（r6）の 4 つを満たしており，ユーザが要求するファイルアクセス手法に最も近い．他の分散ファイルシステムはカーネルモードで動作するのに対し，SFS はオペレーティングシステムに組み込まれている NFS の通信をユーザモードで中継する方式をとっている．このため，グリッドでは多種のオペレーティングシステムが混在するが，多くの計算リソース上で SFS を動作させることができる．新規に 6 つの要求を満たすソフトウェアを開発することには長期間を要するため，要求（r4）と（r5）を満たすように SFS を拡張することがユーザの要求を満たすための最善のアプローチであると考えられる．

3.4 提案手法と実装

本研究では，3.2 節で挙げた 6 つの要求を満足するファイルアクセス手法を提案する．以下では，提案手法とその特徴について述べ，SFS に新たに追加したプログラムの実装を説明する．

3.4.1 提案手法の概要

本提案手法は，SFS の持つ高い位置透過性を保ちながら，グリッド認証基盤に適応しシングルサインオンで利用できる点と，委譲証明書の受け付けを制限することで委譲証明書が盗用されるリスクを低減できる点を特徴として有する．シングルサインオンの実現のために，SFS では利便性を損なう要因の 1 つであった鍵の登録を，SFS の self-certifying pathname 検索機能と GSI の認証と暗号化の機能を応用して，安全かつ自動的に行うメカニズムを提案する．また，委譲証明書が盗用されるリスクを低減するために，受け付けを許可または拒否するプロキシ証明書の識別名を GSI の機能を応用してサーバに送信し，ユーザが遠隔設定する機能の実現方法を提案する．

以下，本提案手法の実装を GSI-SFS と呼ぶ．図 3.1 にその GSI-SFS のアーキテクチャを示す．図に示すように，SFS サーバと SFS クライアントはオペレーティングシステムの NFS 通信を暗号化して中継する機能を持つ．その際，認証には公開鍵暗号が使用され，SFS サーバと SFS クライアントにそれぞれ対称となる鍵が必要である．本研究では，この鍵の登録を GSI の機能を応用して自動化するために，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）を新たに開発した．これらのプログラムはユーザが委譲証明書の制限を設定するためにも用いられる．以下では，グリッド認証基盤への適用と，委譲証明書の制限の，それぞれの実現手法について説明した後，提案手法の評価を行う．

3.4.2 グリッド認証基盤への適応

要求（r4）のグリッド認証基盤への適応を実現するためには，まず SFS のソースコードを変更し認証部分を GSI で置き換えることが考えられる．しかし，SFS は公開鍵のハッシュ値を self-certifying pathname に埋め込むなど強く独自仕様の公開鍵に依存しており，SFS の認証を GSI で置き換えることは難しい．そこで本研究では，GSI を用いてこの公開鍵の登録を自動化することでシングルサインオンを実現し，要求（r4）を満たすことを考える．

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

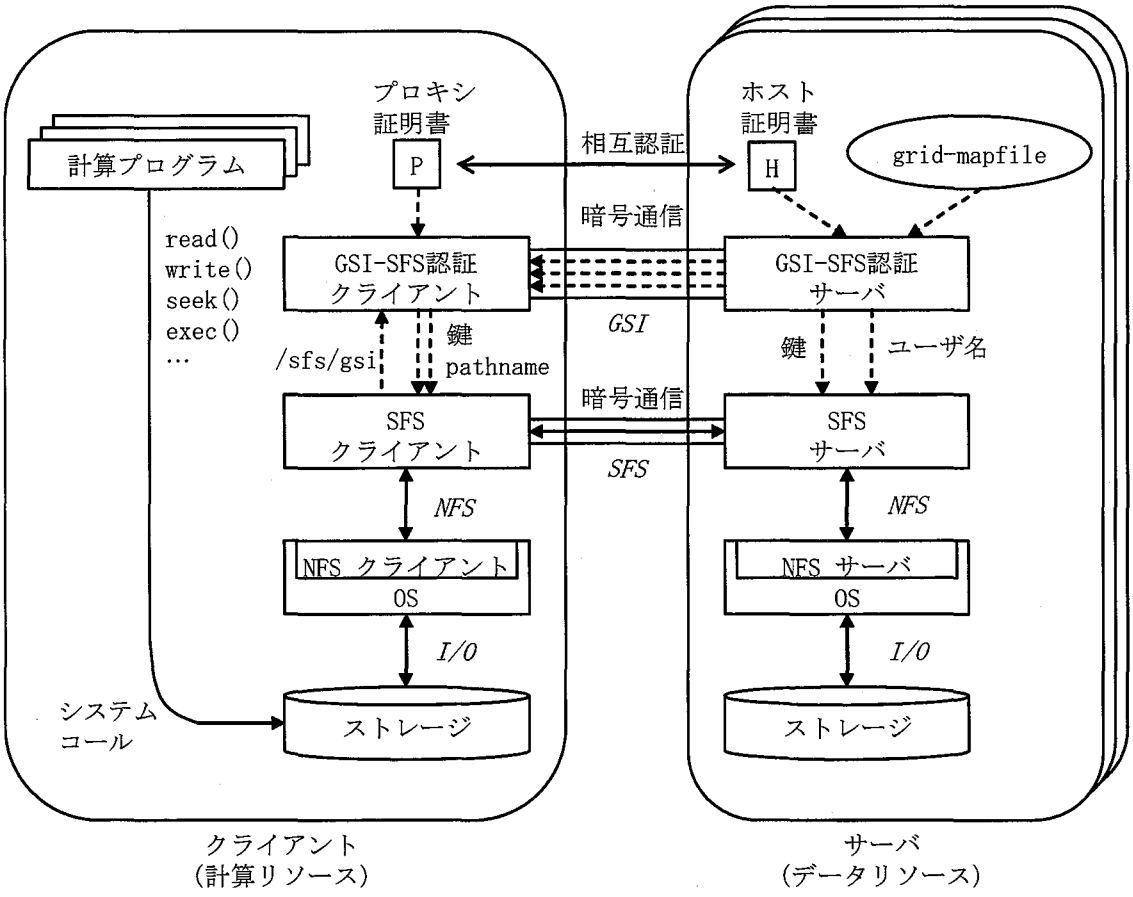

図 3.1 GSI-SFS のアーキテクチャ.

備わっている．SFS クライアントは，/sfs/の直後に@以外の文字が続くディレクトリ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．GSI-SFS 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 の認証を実現した．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，3.3.3 節で説明した SFS の *HostID* を使用せずに一意なファイル名空間を実現している．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計算プログラムによって多用されるが，ホームディレクトリのパスはリソースによって異なる．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可能である.

SFS を利用するためには, SFS サーバと SFS クライアントにそれぞれ公開鍵暗号の対称な鍵ペアが登録されていることが必要である. これらの鍵を自動的に登録するには, クライアントで鍵ペアを生成してサーバに片方を送信する方法と, サーバ側で鍵ペアを生成してクライアントに片方を送信する方法の, 2 種類の方法が考えられる. 鍵の生成では推定されにくい疑似乱数を生成する必要があり数秒の時間を要するため, 鍵登録に要する時間を小さくするためには鍵の生成回数を必要最小限にする必要がある. そのためには, 繰り返し同じサーバにアクセスする際には一度登録した鍵を再利用することが考えられるが, クライアントで鍵ペアを生成する前者の方法ではサーバに鍵の再利用が可能であるか問い合わせる必要があり, 非効率である. この理由から, GSI-SFS では必要に応じてサーバ側で鍵ペアを生成し, クライアントに片方を GSI の機能で暗号化して送信する方法を採用する.

GSI-SFS での認証の流れを以下に示す.

1. 計算プログラムが/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を取得し, 認証サーバはリソース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 2.2.3 節で説明した grid-mapfile を参照してユーザ証明書の識別名とオペレーティングシステムの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の識別名から知ることができる.
5. 認証サーバは対応づけられたユーザのための鍵ペアを生成し, 片方の鍵を 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鍵, SFS サーバの *HostID*, サーバ上にあるユーザホームディ

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．鍵と *HostID* は SFS サーバに接続するために最低限必要な情報である．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，登録されている鍵を使用して SFS サーバに接続する．

以上により，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る．

3.4.3 委譲証明書の制限

要求（r5）の委譲証明書の制限を実現するためには，利用ポリシを証明書に記述する方法と，別途記述する方法の 2 つが考えられる．前者ではポリシと，そのポリシがどのポリシ記述言語で記述されているかを格納するプロキシ証明書のフィールドを利用する必要がある．これを利用するには以下の手順を踏む必要があると規定されている [18].

1. ポリシ言語を定義する．
2. ポリシ言語を規定の手続きを経て登録する．
3. プロキシ証明書を受け取る全てのサービスがポリシ言語を理解できるようにする．

利用ポリシを証明書に記述する方法の採用は，多大なる開発コストが問題となる．例えば，ポリシ言語を厳密に定義して文書化し，事務的な手続きを行うことによる大きな開発コストが発生する．さらに，3 はグリッド上のプロキシ証明書を受け取る全てのサービスを拡張してポリシ言語を理解できるようにする必要がある．そのため，この方法は現実的には困難である．一方，ポリシを別に保存する方法は比較的導入が容易である．例えば，プロキシ証明書の識別名が一意であるという特性を利用すればポリシを必ずしも証明書に埋め込む必要がなく，別に記述しても対応付けることが可能である．本研究で必要とするポリシを別に記述する場合，認証サーバに何らかの方法でポリシを転送しなければならない．ここでポリシが改ざんされると意味をなさないため，ポリシは暗号化する必要がある．GRAM の機能を利用してポリシファイルを GSI で暗号化して伝播していくことも考えられるが，このポリシは SFS サーバにしか必要のない情報である．この理由から，ポリ

シはGRAMを用いず，GSIで暗号化して直接，認証サーバに送信する方法が確実かつ簡潔であると考えた．

委譲証明書の制限は，要求(r4)を満たすために開発した認証クライアント(gsisfskey)と認証サーバ（gsisfskeyd）に機能追加を行うことで実現した．認証クライアントには--authorizeと--unauthorizeオプションを追加し，これを用いてユーザが委譲証明書の許可または拒否の設定を行えるようにした．認証サーバには設定ファイルにこの機能を有効にするオプション項目を追加した．

委譲証明書の制限が有効にされた認証サーバで，委譲証明書でのアクセスを許可する手順を図3.2に示す．計算リソース上のプロセスが委譲証明書を使って認証サーバにアクセスすることを許可するには，まずユーザはgsisfskeyを次のように使用してSFSサーバ上にプロキシ証明書の識別名を登録する（図3.2-1）．

gsisfskey --authorize *hostname*

gsisfskeyは--authorizeオプションを付けて実行されるとプロキシ証明書を取得する．それが委譲証明書であればエラーを表示して終了し，そうでなければgsisfskeydとGSIでの認証を行う．gsisfskeydはプロキシ証明書が委譲証明書であるかどうか確認し，委譲証明書でなければ従来と同様にSFSを利用するための情報を送信する．委譲証明書であれば，プロキシ証明書の識別名がユーザのホームディレクトリ内の.gsisfs/authorized_proxy（図3.3）に登録されている識別名で始まっているかを確認する．始まっていればSFSの情報を，そうでなければ文字列DENYを送信する．gsisfskeyはDENYを受信すると文字列OKを返信して終了する．SFSの情報を受信した場合，この場合--authorizeを付けて実行されているため文字列AUTHORIZE証明書の期限を返信して終了する．gsisfskeydはOKを受信するとそのまま終了するが，AUTHORIZEを受信した場合は.gsisfs/authorized_proxyにプロキシ証明書の識別名と期限を追記し，期限の切れた識別名があれば削除して終了する．期限はUNIXで一般的に使われる1970年1月1日0時0分0秒UTCからの経過時間で表されており，単位は秒である．セキュリティ上の期限の確認はGSIの認証で行われ，ここに保存している期限はファイルが肥大化することを防ぐためのものである．登録が完了すると，ユーザやグリッドポータルはGRAMを用いて計算リソースにジョブの投入を行う（図3.2-2）．このとき，委譲証明書の識別名はプロキシ証明書の識別名にさらに一意な番号を付加したものになる．計算リソースがGSI-SFSサーバ上のファイルにアクセスする際にはこの委譲証明書が使用される（図3.2-3）．GSI-SFSサーバはクライアントの委譲証明書の識別名が登録されている識別名で始まっているかを確認する（図3.2-4）．始まっていればクライアントにSFSの情報を，そうでなければDENYを送信する．

すでに登録されている識別名を削除するには以下のようにgsisfskeyを使用する．

gsisfskey --unauthorize *hostname*

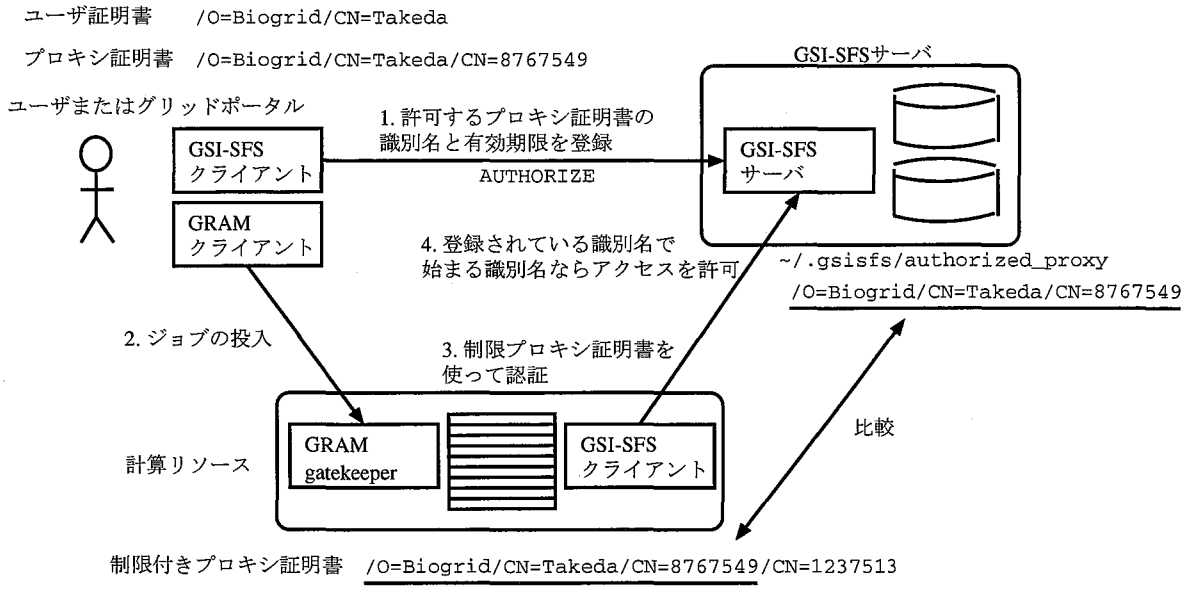

図 3.2 委譲証明書を用いたサーバへのアクセス制限.

```
/O=Grid/CN=Shingo TAKEDA/CN=242970403 1107712175
/O=Grid/CN=Shingo TAKEDA/CN=139637136 1107716444
```

図 3.3 authorized_proxy ファイルの例.

この場合, `--authorize` と同様の動作をするが, `AUTHORIZE` ではなく `UNAUTHORIZE` を返信する. これを受信した `gsisfskeyd` は該当する識別名をファイルから削除する.

この機能によって, 委譲証明書で GSI-SFS を利用する場合は, あらかじめユーザによって許可された SFS サーバにしかアクセスできない. これにより, もし権限委譲したコンピュータで問題が発生し, 委譲証明書が悪用されても, 不正アクセスの被害を一部の SFS サーバだけに抑えることができる.

3.5 評価

GSI-SFS は, 科学技術計算のファイルアクセスにおいて, 安全性と利便性を両立することを目標に開発した. しかし, 計算プログラムには高いデータ入出力効率を要求するものもあるため, データ転送のスループットについても評価する必要がある. 本節では, まず GSI-SFS のスループットを評価し, 科学技術計算への有効性を議論する. 次に, GSI-SFS の利用例を想定し, 3.2 節で挙げたユーザの安全性と位置透過性に関する 6 つの要求を満

表 3.2 測定に使用した PC の仕様.

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 認証サーバ/クライアント	0.0.7
NISTNet	2.0.12
Netperf	2.2pl2

足していることを確認する.

3.5.1 スループット

ここでは適用可能な科学技術計算を考えるために，GSI-SFS のスループットを評価する．SFS は UDP の NFS を暗号化し，TCP で中継する仕組みになっている．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 を前提に設計されており，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 のスループットと，比較対象として GridFTP のスループットの測定を行った．インターネットでは回線の状態が常に変化しているため測定結果の厳密な比較が行えない．そこで Linux 用の WAN エミュレータである NISTNet [56] を使用して，LAN 上で遅延や帯域をエミュレートして測定を行った．測定では表 3.2 に示す同じ仕様の 2 台の PC を HUB を介して接続した．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 バイト，送信バッファサイズは同様に 4,096/16,384/131,072 バイト，ウインドウスケーリングは有効である．

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 3.4 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 3.5 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 Netperf [57] で測定した値である．図 3.4 では 160 Mbps 程度，図 3.5 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れる．このような遅延が非常に小さい LAN のような環境では，安全性と性能がトレードオフの関係になると言える．

次に帯域を 1,000 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 3.6 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 3.7 に示す．図 3.6 と図 3.7 の両方において RTT が大きくなるにしたがって 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 および 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．一方 GridFTP は複数の TCP ストリームで並列転送を行っているため，遅延の大きいネットワークにおいても効率の低下が小さい．

以上の実験結果より，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わかる．すなわち，GSI-SFS は 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しない計算プログラム，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．文献 [58] によると代表的な科学技術計算プログラムのうち SETI@home，バイオインフォマティクスの BLAST，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 は CPU の負荷が高いが計算時に発生する I/O トラフィックは 10 kbps から 10 Mbps 程度までと小さいとされており，GSI-SFS はこのような特性の計算プログラムに適していると考える．

3.5.2 安全性と位置透過性

本研究では，科学技術計算におけるユーザの利便性向上を目的とし，既存の分散ファイルシステムをグリッド認証基盤に適応させることで，3.2 節に記したユーザの 6 つの要求を満たすファイルアクセス手法を提案した．本節では，科学技術計算における GSI-SFS の利用例を示し，安全性と位置透過性を評価する．利用例としては，ユーザが計算プログラムを作成し，データリソース上のデータファイルを複数の計算リソースを使用して処理する典型的な場合を考える．図 3.8 に示す例では，リソース A に計算プログラムと出力ファイル，リソース D に入力ファイルがあり，リソース B と C は計算能力を提供する．

GSI-SFS を利用すれば，このジョブは図 3.9 のように記述することができる．ユーザは計算プログラム（`calc.exe`）の作成においてユーザは POSIX API を利用することができ，シングルサインオンのために Globus Toolkit のライブラリを用いて計算プログラムを

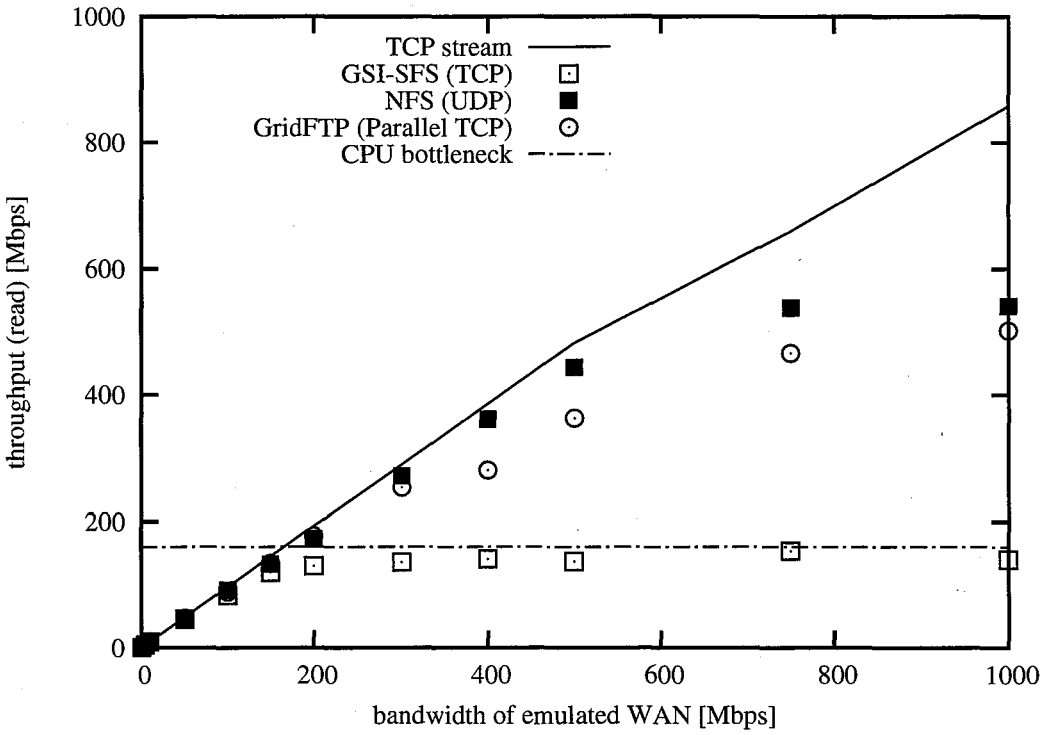

図 3.4 遅延の小さい環境での帯域と読み込みスループット.

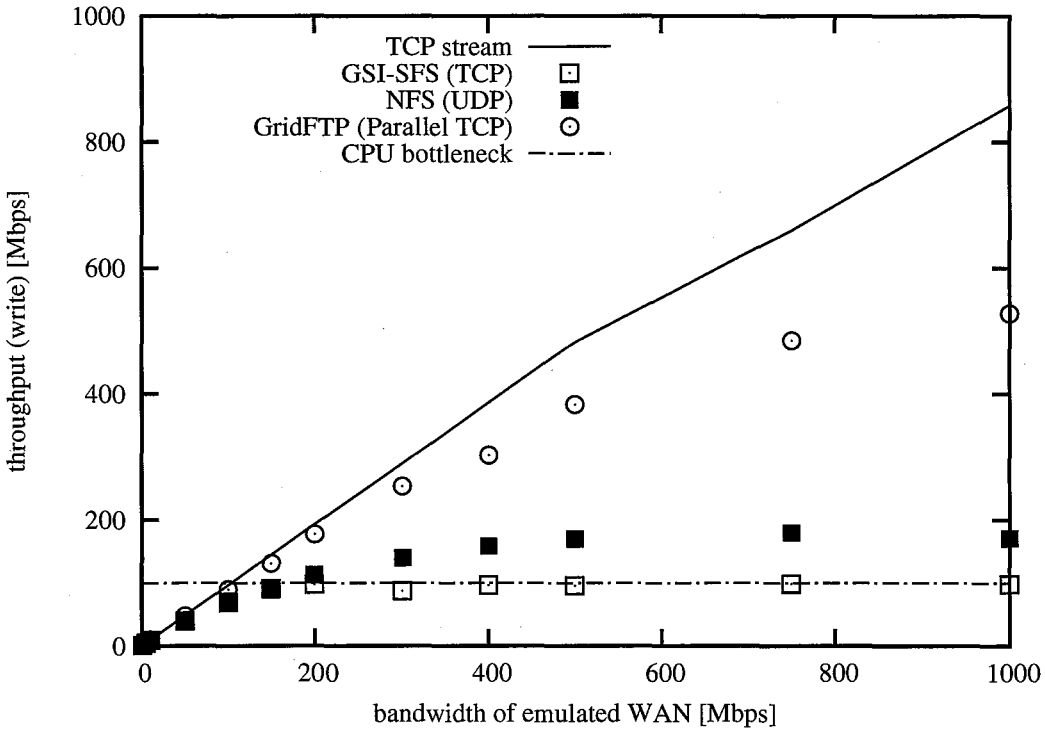

図 3.5 遅延の小さい環境での帯域と書き込みスループット.

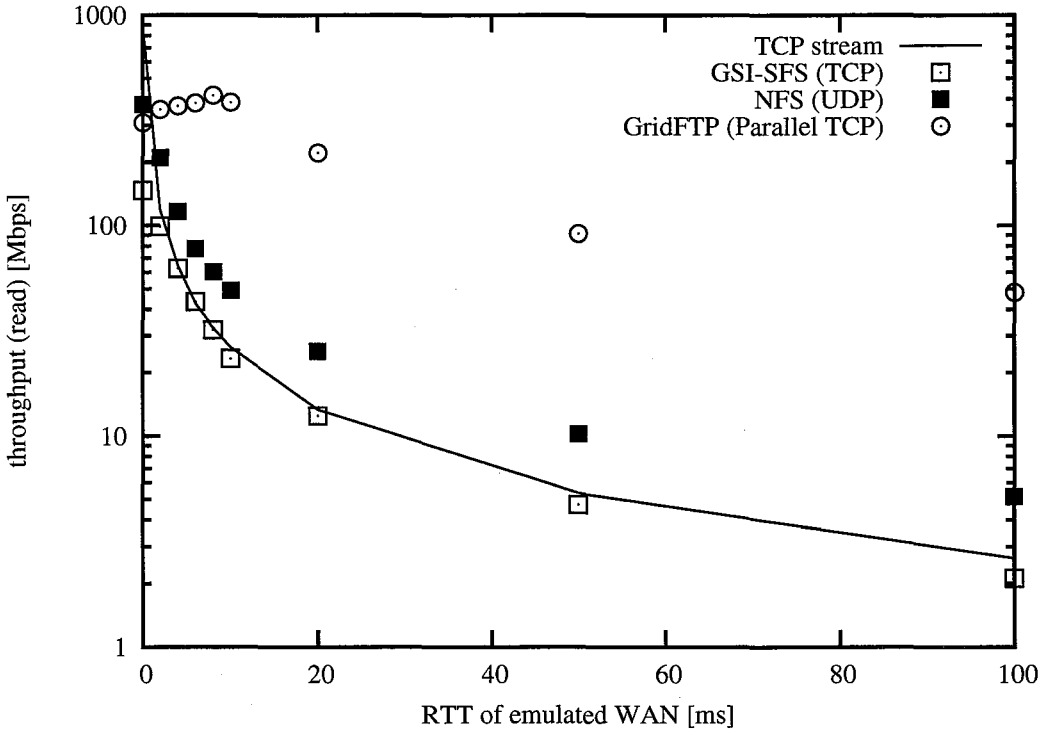

図 3.6 遅延の大きい環境での RTT と読み込みスループット.

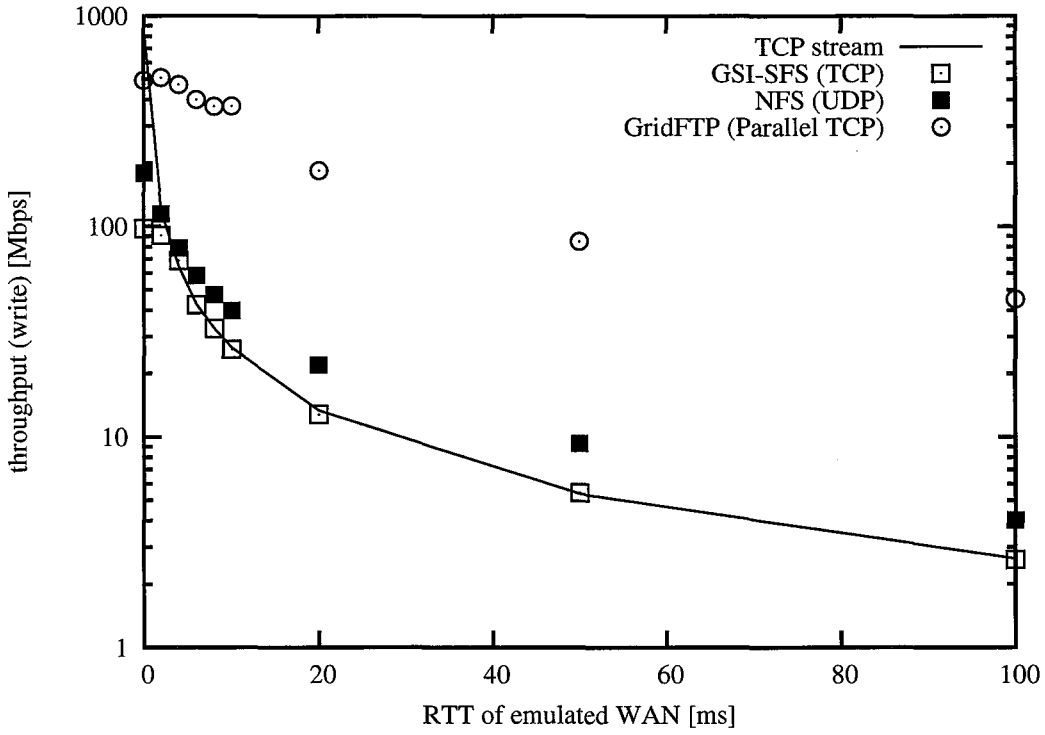

図 3.7 遅延の大きい環境での RTT と書き込みスループット.

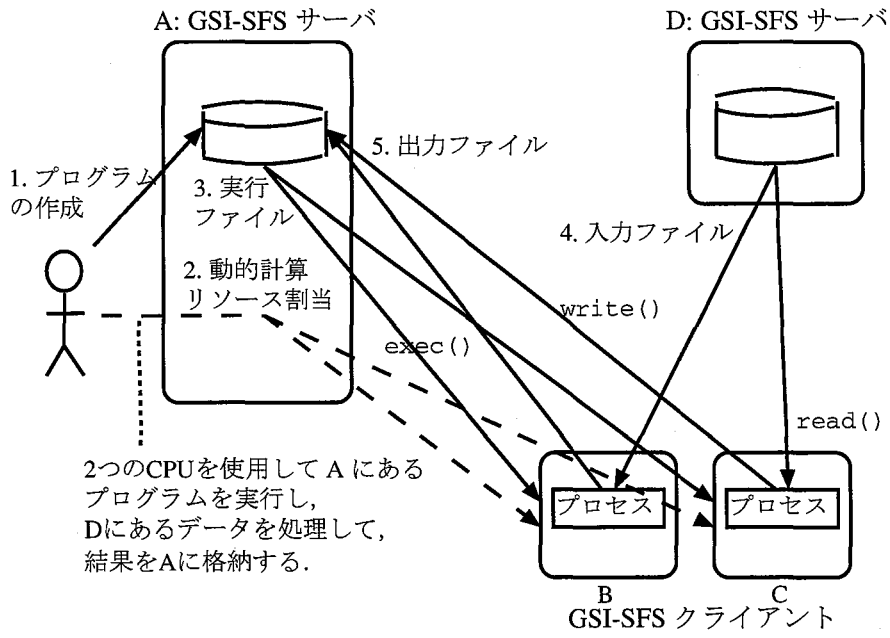

図 3.8 グリッド上のデータの分散並列処理.

拡張する必要がなく，要求（r1）が満たされている．計算プログラムは実行するリソースにコピーすることなく実行でき，要求（r2）が満たされている．GSI-SFS でも SFS と同様に一意なファイル名空間を実現しており，どの計算リソースでこのジョブが実行されても同一のファイルにアクセスすることができ，要求（r3）が満たされている．2.4.1 節では，提案手法を使用せずにシングルサインオンを実現する以下の 3 つの方法を挙げた．

1. Globus Toolkit の API を使用した計算プログラムの拡張
2. Globus Toolkit のコマンドを使用した入力ファイルの計算前転送
3. 遠隔ファイルシステムの静的マウント

第 1 の方法では，計算プログラムの拡張が必要である．本システムの GSI-SFS 認証クライアント（`gsisfskey`）は GSI の機能を使用してデータを転送する単純なプログラムの例であるが，C 言語のソースコードの行数は 764 であり，ソケットの生成や例外処理が多くを占める．このようなプログラムは Globus Toolkit の詳細な仕様を知らなければ記述することができず，科学技術計算を目的としたユーザが記述する必要があると利便性を大きく損ねる．第 2 の方法では，シェルスクリプトでファイルを転送するためプログラムの拡張は不要である．同様の処理を実現するスクリプトは図 3.10 のようになる．この方法では，提案手法では 1 ステップであった処理が 4 ステップになり記述が複雑になるだけでなく，入力ファイルの転送が完了するまで計算が開始できず非効率である．また，計算リソースに一時ファイルを保管する領域が必要となる問題も発生する．第 3 の方法では，管

```
/gsisfs-home/hostname.of.A/calc.exe --param $PARAM \
   --input-file /gsisfs-home/hostname.of.D/input.dat \
   --output-file /gsisfs-home/hostname.of.A/output.dat
```

図 3.9 GSI-SFS を用いたジョブの記述.

```
globus-url-copy gridftp://hostname.of.A/home/shingo/calc.exe \
                file:///home/takeda/calc.exe
globus-url-copy gridftp://hostname.of.D/home/stakeda/input.dat \
                file:///home/takeda/input.dat
/home/takeda/calc.exe --param $PARAM \
        --input-file input.dat --output-file output.dat
globus-url-copy file:///home/takeda/output.dat \
                gridftp://hostname.of.A/home/shingo/output.dat
```

図 3.10 シェルスクリプトを用いたジョブの記述.

```
/mnt/hostA_homedir/shingo/calc.exe --param $PARAM \
   --input-file /tmp/data_on_hostD/home/stakeda/input.dat \
   --output-file /mount/hostA_homedir/shingo/output.dat
```

図 3.11 静的マウントを用いたジョブの記述.

理者があらかじめファイルシステムをマウントしていることが前提となる．どのディレクトリにマウントするかはリソースの管理者が決定するため，リソースごとにディレクトリ構造は異なる．この場合，ジョブの記述は図 3.11 のようになるが，ファイル名の一意性が確保されていないため，同じ記述で異なるリソースでも動作するとは限らない．GSI-SFS はこれらの 3 つの方法で発生する問題を解決しており，利便性の面で優れている．

GSI-SFS では，プロキシ証明書を使用してシングルサインオンが実現されており，ユーザが認証を意識する必要はなく，要求（r4）が満たされている．機密データが保存される可能性のある SFS サーバでは委譲証明書の制限機能を有効にし，許可された委譲証明書のみ受け付けることができる．この場合，図中 B または C 上に生成されている委譲証明書が万一盗用されても，他のこの設定が有効な SFS サーバにはアクセスできず，要求（r5）が満たされている．以上の認証，認可，及び，データ転送において全ての通信データは GSI と SFS の機能によって暗号化され，要求（r6）が満たされている．以上から，GSI-SFS では安全性を損なうことなく利便性を高めている．

3.6 グリッドテストベッドへの応用事例

本節では，GSI-SFS の実用性を示す応用事例として，大阪大学と中国の中国科学院を接続したグリッドテストベッドについて述べる．中国には多種の生物が棲息しており，その生物種は地球上の生物種の 70% にも及ぶ．中国科学院は多数の中国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．大阪大学には高性能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 GSI-SFS を用いて安全かつ位置透過的に接続することで，中国の貴重な生物データの高速な解析が可能となり，生物学の発展が期待される．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．以下ではこのグリッドシステムの詳細について述べる．

3.6.1 科学者の要求と設計

システムのユーザとして想定するのは製薬会社の研究員などであり，これらのユーザはグリッドに関する深い知識を有していないのが一般的である．そこで直感的に利用できるグラフィカルユーザインタフェースが必要である．また実際のデータがどのデータリソース上にあり，計算がどの計算リソース上で行われるかを意識させないためには，シングルサインオンが必須となる．さらにこれらの分野では機密情報の漏えいによるリスクが非常に大きいため，適切にアクセス制御を行いデータ機密性を保護することが求められる．対応させる計算プログラムとしては，生物情報学の分野で広く利用されている BLAST と ClustalW を採用した．

グラフィカルユーザインタフェースには，Web インタフェースを採用した．Web インタフェースを提供することで研究者は自身のクライアントにソフトウェアを追加インストールする必要がなく，Web ブラウザから機能を利用できる．計算リソースへの計算要求の送信には 2.3.2 節で説明した Globus Toolkit のサービスの 1 つである GRAM を用いる．BLAST と ClustalW の入出力には GSI-SFS を用いた．3.6.4 節に記したように SFS のスループットは 10 Mbps 程度であるが，BLAST と ClustalW ではデータ転送に比べて CPU 負荷の高いため，GSI-SFS によるデータ転送は十分実用的であると考えられる．GSI-SFS では計算プログラムの変更を不要でこれらの計算プログラムをそのまま利用でき，また新たな計算プログラムを追加することも容易である．この Web インタフェースから GRAM の API を呼び出すためには Java COG Kit を使用する．

表 3.3 生物データを提供するデータリソースの仕様.

CPU	PentiumIII 800MHz
メインメモリ	154MB
ハードディスク	Ultra ATA 100 160GB
ネットワーク	100BASE-T
RedHat Linux	8.0 (Kernel 2.4.18)
Globus Toolkit	2.0
SFS	0.7.2
GSI-SFS 認証サーバ/クライアント	0.0.6a

3.6.2 グリッド環境の構成

構築したグリッドシステムの構成を図 3.12 に示す．中国科学院の生物データは表 3.3 に示す仕様の SFS サーバで提供される．データ解析に利用した大阪大学バイオグリッド基盤システムは表 3.4 に示すブレードサーバ（計算リソース）が 78 台ネットワークで接続されており，それぞれが CPU を 2 基ずつ搭載しているため最大計 156 CPU が利用できる．

このグリッドシステムの主なユーザは生物学者や製薬会社の研究員である．各計算リソースには，これらのユーザがよく利用する BLAST と ClustalW がインストールされている．BLAST は日本電気株式会社によってクラスタ向けに並列化されたものが導入されており，複数の計算リソースで並列に解析を行える．ClustalW は大きな計算能力を必要としないため並列化されておらず，単一 CPU でのみ動作させる．各計算リソースは GSI-SFS 認証クライアントとして機能し，計算プログラムは GSI-SFS 認証サーバに接続して中国科学院の生物データにアクセスする．中国科学院にはこのグリッドシステムを利用するためのグリッドポータル Grid User Interface to the Distributed Environment (GUIDE) [59] が設置されている．GUIDE は Java Servlet を用いて開発されており，ユーザに Web インタフェースを提供する．ユーザは Web ブラウザから BLAST と ClustalW を利用することができる．GUIDE では Java と Globus Toolkit との連携部分に Java CoG Kit を用いている．

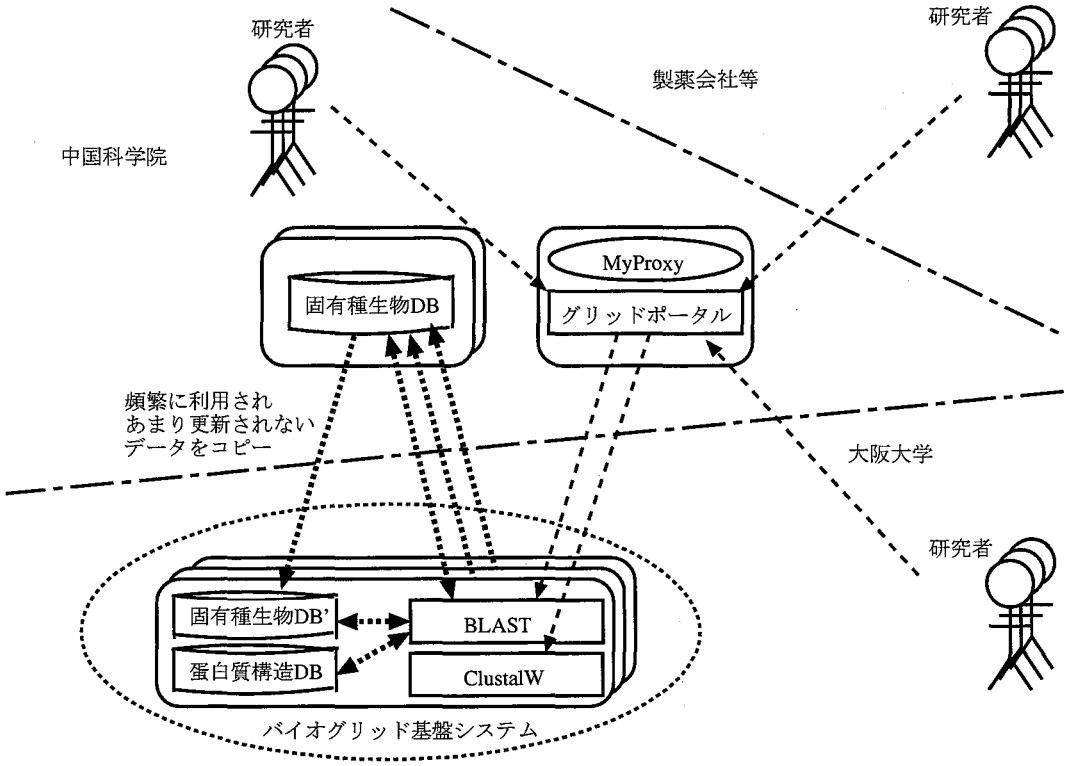

図 3.12 生物情報を対象としたグリッドテストベッド.

表 3.4 データ解析に利用したシステムを構成するブレードサーバの仕様.

CPU	PentiumIII-S 1.4GHz × 2
メインメモリ	1GB
ハードディスク	Ultra ATA 100 160GB
ネットワーク	100BASE-T × 6
RedHat Linux	7.3 (Kernel 2.4.19 with SCore)
Globus Toolkit	2.0
SFS	0.7.2
GSI-SFS 認証サーバ/クライアント	0.0.6a

3.6.3 シングルサインオン認証

計算ノードにジョブを投入するには GRAM を使用する．GRAM の認証サービスである gatekeeper は GSI を用いているため，ユーザ証明書とそれに対応する秘密鍵で認証してジョブを投入する必要がある．このとき，GUIDE がユーザに代わってジョブの投入を

行うが，ユーザ証明書と秘密鍵をグリッドポータルの Web サーバに保存しておくと，それらが不正利用された場合に大きな被害を受ける可能性がある．このため，GUIDE では MyProxy [37] 証明書リポジトリを使用している．ユーザは GUIDE にログインする前に，MyProxy にプロキシ証明書を保存しておく．プロキシ証明書はユーザ証明書の期限と比べて期限が短く，不正利用されたときのリスクを低減することができる．

ユーザが GUIDE のログインページで正しいユーザ名とパスワードを入力すると，GUIDE が MyProxy からユーザのプロキシ証明書と秘密鍵を取得し，BLAST のページに切り替わる．ユーザはデータを処理する計算リソース（現在は大阪大学のみ），検索対象のデータベース，検索する配列，その他のオプションをここで選択できる．ユーザがジョブの投入ボタンをクリックすると，GUIDE はユーザのプロキシ証明書と秘密鍵を使用して計算リソースの gatekeeper と認証を行い，計算プログラムを実行する．このとき GUIDE から計算リソース上のプロセスに委譲証明書が発行される．計算プログラムは委譲証明書を用いて GSI-SFS 認証サーバと相互認証を行う．ユーザは大阪大学と中国科学院の多数のリソースを利用するが，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．計算の要求後はジョブ管理ページに切り替わり，ユーザは計算の状態（実行中，エラー，完了）を確認することができる．計算が完了すると結果ページに移動できるようになる．ユーザは BLAST の結果を ClustalW の入力とすることが多いため，BLAST の結果を選択して ClustalW をすぐ利用できるようにインタフェースが設計されている．

このように，GSI-SFS と，GUIDE，GRAM，MyProxy を組み合わせることによってシングルサインオン認証でユーザが中国科学院と大阪大学にある計算リソース及びデータリソースを安全かつ位置透過的に利用できるグリッドを構築できる．

本テストベッドでは GSI-SFS 認証サーバの委譲証明書を制限する機能は使用していない．中国科学院が提供している生物データは機密性の高いものではなく，ユーザがこのサーバに機密性の高いファイルを保存することも無いためである．今後，テストベッドの拡張によって機密性の高いデータを扱う場合は，本研究で実装した委譲証明書を制限する機能を使用し，安全性を高めることが可能である．

3.6.4 解析における処理性能

BLAST は DNA 配列などの相同性検索プログラムの 1 種であり，データファイルに含まれる配列から検索対象の配列と類似したものを検索する．この処理では，データファイルに逐次アクセスを行い，ランダムアクセスは発生しない．そのため，逐次アクセスにおけるスループットが処理性能に大きく影響する．ここでは本テストベッドにおいて処理性能を計測した結果について述べ，議論を行う．

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 つの計算リソースで BLAST の検索処理を行った場合，1 時間程度を要した．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 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，頻繁に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計算リソースに手作業でコピー（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 台の計算リソースで 1 分，32 台で 30 秒程度で処理することができた．

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も検討している．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．

3.6.5 セキュリティに関する知見

本節ではテストベッド実験から得られたセキュリティに関する知見を述べる．本テストベッドにおいて中国科学院が提供している生物データの機密性は低い．しかし，製薬会社の研究員などにとっては，それが公開データであっても，どのデータを分析しているのかという情報が他ユーザに対して機密情報となる．このため，このようなユーザは本テストベッド実験を通じて，どのファイルにアクセスしているかを他ユーザに知られないことを強く要求することがあった．

そのためには，まずネットワーク上で盗聴を防止する必要がある．このテストベッドにおいて発生する通信は以下の 3 つである．

1. ユーザの操作するクライアントと GUIDE サーバの間
2. GUIDE サーバと計算リソースの間
3. 計算リソースとデータリソース（SFS サーバ）の間

1 の通信は SSL によって，2 と 3 の通信は GSI によって暗号を使って認証し，安全な通信チャネルを確保したうえでデータの転送を行う．したがって本システムではネットワークを流れる全てのデータが暗号化，検証され，盗聴や改ざんからデータが保護されている．

次に，グリッドでは計算リソースが複数のユーザによって共有されるため，同じリソースを利用している他ユーザからどのデータを分析しているかを知られない必要がある．SFS では各ユーザは図 3.13 に示すような排他的なディレクトリ構造を持つが，これは

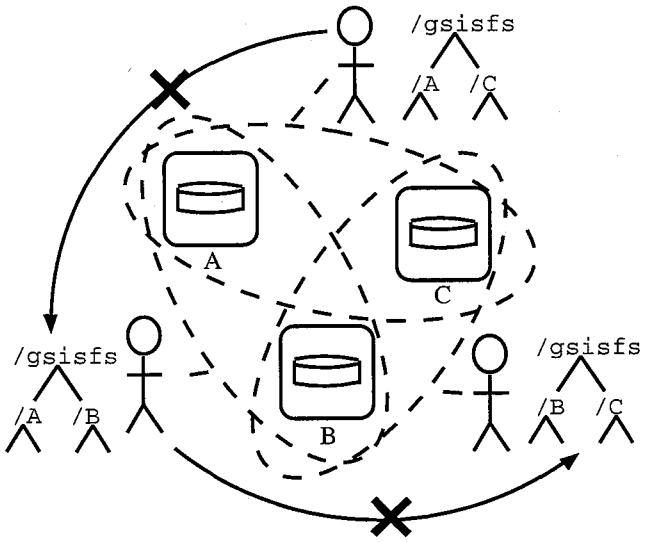

図 3.13 排他的なディレクトリ構造.

GSI-SFS でも同様である．そのため，計算リソースを複数のユーザで共有した場合においてもユーザは他ユーザのディレクトリ構造は見えず，他ユーザがどのデータを解析しているのか知ることはできない．このように，GSI-SFS を用いることでユーザが他ユーザの処理内容を知ることを防止でき，グリッドにおけるファイルアクセスでは排他的なディレクトリ構造を提供する設計とすることが望ましいという知見が得られた.

3.7 むすび

本章では，既存の分散ファイルシステムをグリッド認証基盤に適応させ，安全性と位置透過性を両立するファイルアクセス手法を提案した．本研究で実装した GSI-SFS は，高い位置透過性を持つ既存の分散ファイルシステム SFS を，グリッド認証基盤 GSI の認証で利用可能にすることで，安全性と位置透過性を両立した.

SFS を GSI で利用可能にするために，GSI の機能を応用して SFS を拡張する GSI-SFS 認証サーバと GSI-SFS 認証クライアントを実装し，それぞれ SFS サーバと SFS クライアントに設置した．GSI-SFS では，ファイルアクセス要求発生時に，SFS の certprog 機能を応用して GSI-SFS 認証クライアントを呼び出し，GSI-SFS 認証クライアントと同サーバ間で GSI の機能で認証を行った後，サーバ側で生成した対称鍵ペアの片方をクライアントに GSI の機能で暗号化して転送する．これにより，サーバとクライアントに SFS の利用に必要な鍵が登録され，シングルサインオンでの利用が可能となる．この鍵登録の過程をユーザが意識する必要はなく，ユーザはファイルの位置を意識することなくグリッドに既存の計算プログラムとデータを安全に展開することが可能となった.

また，本研究では，シングルサインオンのためにリソース上に生成される平文の秘密鍵を含む委譲証明書が漏えいした際のリスクを低減するために，委譲証明書で利用できるサーバを制限する機能をGSI-SFS認証サーバと同クライアントに実装した．ユーザはGSI-SFS認証クライアントを用いて，どの委譲証明書の受け付けを許可または拒否するかの情報をGSI-SFS認証サーバに送信し，遠隔設定することが可能となった．

GSI-SFSでは，GSIとSFSの機能を利用して安全な認証とデータ転送を実現している．しかし，認可の機能に関してはGSIの単純なマッピング機能を使用しており，ファイル単位でのアクセス制御はオペレーティングシステムに依存している．このため，アクセス制御のためには`grid-mapfile`を参照してユーザの識別名とオペレーティングシステム上のユーザ名を対応づけながらアクセス権を設定しなければならない．そのため，特にユーザ数の多いグリッドでの利用には課題が残る．今後，OGFで行われているデータグリッドのアクセス制御に関する研究の動向を参考にしながら，ユーザの識別名を直接指定できるアクセス制御の機能を取り入れていくことが必要である．

第 4 章

リソース要求フローに着目したセキュリティモニタリング

4.1 まえがき

近年では不正利用や機密漏えいを防止し，グリッドを安全に運用するためのセキュリティ技術の必要性と重要性がますます高まる傾向にある．しかし，2 章に記したようにグリッド認証基盤上での科学技術計算では，安全性とシングルサインオンの利便性を確保することは難しく，この問題が管理者による安全なグリッド運用の妨げとなっている．このような背景から，グリッド認証基盤に適応し管理者の負担を低減するためのセキュリティ技術が求められている．

グリッドのためのセキュリティ技術には，GSI の上で高度なアクセス制御を実現する Community Authorization Service (CAS) [60] や，容易な権限管理を実現する VOMS [61] など，様々な技術が研究開発されている．グリッドを安全に運用するためには，管理者はこれら個々の技術を導入するだけでなく，不正利用や機密漏えいと関連しうる異常を迅速に発見，分析，対応していくことが必要である．しかし，GSI のシングルサインオン認証は，ユーザによる複数組織に分散したコンピュータリソースの利用利便性を高める一方で，複数組織に連鎖する不正利用や障害と関連する可能性のある異常の管理者による発見を困難にしている．このような複数組織にまたがる異常を発見するためには，管理者がリソース上に出力されているグリッド認証基盤のログ情報を閲覧し，他のリソースのログ情報や他組織のネットワークの情報なども参照する必要があり，この作業が管理者の大きな負担となっている．

本章は，異常の発見と分析を容易にするグリッド認証基盤に適応したセキュリティモニタリング技術を提案し，管理者の負担を軽減することを目的とする．本章で提案するセキュリティモニタリングシステムは，広域に分散するグリッド認証基盤のログファイルに

含まれる情報を収集，蓄積し，各組織の管理者が他組織の情報も参考にしながら効率的に異常を発見，分析することを可能とする．

以下に本章の構成を示す．4.2 節では，本研究で想定する一般的なグリッドの構成及び運用体制を示し，管理者の負担を軽減するためのセキュリティモニタリングシステムに求められる機能の分析を行う．4.3 節では，4.2 節の要求分析の結果に基づいてグリッドで利用されている既存モニタリング手法を調査し，本研究で着目する異常の発見への適用可能性を検討する．4.4 節では，リソース要求フローに着目したセキュリティモニタリング手法を提案するとともに，その設計と実装について説明する．4.5 節では，実装したシステムをグリッドテストベッドに展開して評価実験を行い，得られた結果を示し，その有用性について議論する．

4.2 要求分析

本節では，管理者の負担を低減するために，セキュリティモニタリングシステムに対する要求を分析する．そのために，まず本研究で想定するグリッド環境について述べる．次に，その環境で回避するべきセキュリティ脅威と，回避するために発見すべき事象について考察する．その考察を基に，セキュリティモニタリングシステムに求められる機能を導く．

4.2.1 想定するグリッド環境

グリッドは複数の組織がコンピュータの計算能力やストレージなどのリソースを共有する環境であり，それぞれの組織に管理者が存在する．図 4.1 に組織の管理者らが連携してグリッドを運用する環境を示す．管理者は，サービスに割り当てられるホスト証明書の管理，ユーザに割り当てられるユーザ証明書の発行，リソースやネットワークの設定，データのバックアップ，不正利用や各種障害への対処など，組織内での作業を担当する．加えて，管理者は他組織の管理者と連携し，Certificate Authority（CA）証明書の交換，相互運用性の維持，複数組織にまたがる不正利用や障害への対処なども行う必要がある．

これらグリッドを構成するリソースを有する組織の管理者の中には，グリッドの全体的な方針を定める代表的な管理者が存在するのが一般的である．本論文では当該管理者をグリッド管理者と定義する．本研究ではこのような，グリッド管理者が主導し組織の管理者らが連携してグリッドを運用するという形態を想定し，グリッド管理者と組織の管理者らによる異常の発見と分析にかかる負担を軽減するセキュリティモニタリングシステムの提案を行う．

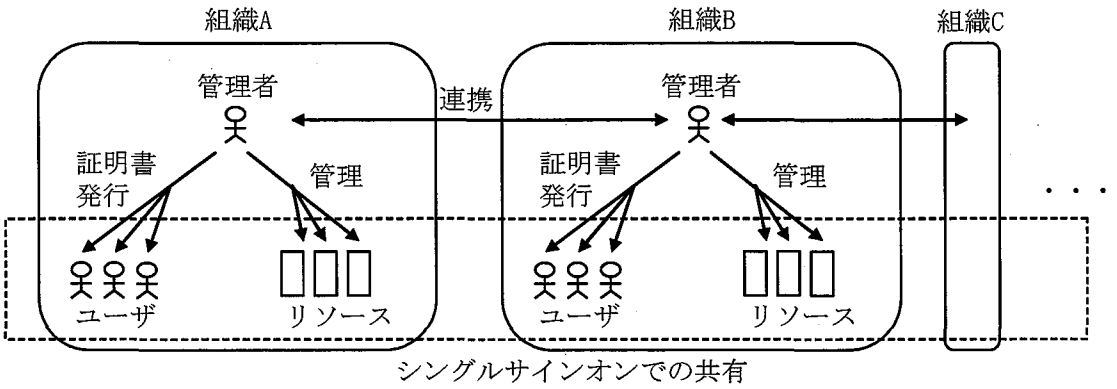

図 4.1 組織の管理者らの連携によるグリッド運用.

4.2.2 セキュリティ脅威とモニタリング対象

グリッドの安全な運用のためには，不正利用，機密漏えい，及び，それらの原因となりうる障害を防止する，またそのような事象の発生を早期発見し被害の拡大を防ぐことが肝要である．グリッドでは，複数リソース上に連鎖的に生成されるプロキシ証明書が万一，ひとつでも盗用されると，当該ユーザの権限ですべてのリソースが不正利用される可能性がある．高性能コンピュータが高速ネットワークで接続された環境で，それらのコンピュータが Distributed Denial of Service（DDoS：分散サービス拒否）攻撃や SPAM の送信に多数悪用されると，ネットワークシステムの機能停止など非常に大きな被害が発生する恐れがある．また，本来のユーザがアクセスできるすべての機密データに漏えいの危険性が生じる．

グリッドの不正利用には，大別して，グリッドを構成する組織外の第三者が行う場合と，グリッドを構成する組織に所属するユーザがその権限範囲を超えて行う場合がある．前者では，組織外の第三者は正規のユーザ証明書を入手することが困難であるため，多数のリソース要求を試行錯誤的に送信し，ソフトウェアの実装不具合や設定誤りを悪用した不正利用を試みる可能性がある．一方，後者ではユーザは自身のユーザ証明書を保持しており，当該ユーザにはあらかじめ許可されたリソースの正当な利用権限がある．2 章で記したように，グリッドではシングルサインオンのためにプロキシ証明書がリソース上に生成される．悪意のあるユーザはこれを利用することを考え，リソース上に生成された他ユーザのプロキシ証明書をソフトウェアの実装不具合や設定誤りを悪用して不正入手する可能性がある．また，廃棄ハードディスクからの入手など，物理的な手段でユーザが所属する組織の他ユーザのユーザ証明書を盗用することも考えられる．

上述した不正利用を早期発見するためには，要求の送信元の正当性をモニタリングする

必要がある．一方，不正利用の兆候を早期発見するためには，侵入試行とソフトウェアの実装不具合や設定誤りといった脆弱性をモニタリングする必要がある．これらを総合的に考え，本研究では次の4つの異常に着目する．

異常（A）グリッドを構成する組織外からのリソース要求

リソースが不正利用されている，または機密が漏えいしている可能性がある．

異常（B）ユーザが所属する組織外からの当該ユーザ証明書の使用

多くの場合，権限委譲やユーザの物理的移動によって発生する正当なリソース要求であるが，ユーザ証明書が他人に不正利用されている可能性がある．

異常（C）大量の認証・認可の失敗

ソフトウェアの実装不具合や，設定誤りを悪用した不正利用を試みられている可能性がある．また，サービスを提供するソフトウェアやハードウェアに障害が発生している可能性もあり，早期に対応する必要がある．

異常（D）ログファイルへの形式外の出力

正常時，ソフトウェアのログファイルはある一定の形式で出力される．例えば，多くのログファイルでは行頭に時刻が出力され，そのあとに処理結果が続く．この形式に従わない出力がなされた場合，ソフトウェアの実装不具合や，設定誤りが存在すると考えられ，不正利用につながる可能性がある．

本章では，以上4つの異常を，管理者が容易に発見，分析することを可能とするセキュリティモニタリングシステムの研究開発を行い，管理者がグリッドを安全に運用する上での利便性に関する問題の解決を目指す．

4.2.3 機能要件

4.2.2節に記した4つの異常を発見するためには，管理者はサービスを提供するソフトウェアのログファイルを調査する必要がある．一般的に，ログファイルはテキスト形式で，リソース要求元のユーザ識別名，クライアントIPアドレス，認証・認可の成否などを含んだ詳細な情報が，時系列に記録される．このようなログファイルのみを管理者が詳細に調査し，上述した異常を発見することは非常に困難である．グリッドでは，ユーザはシングルサインオンで複数のリソースを利用でき，多くのリソースを認証を意識せずに使用して作業を効率化できる．しかし，実際にはリソース要求が発生するたびにシングルサイオンを実現する認証・認可処理が対象となるリソース間で行われており，それら該当するリソースの生成するログファイルには先述した詳細な情報が毎回記録されている．そのため，グリッドの安全な運用のためには，管理者はこれらログファイルを注意深く調査する必要があるが，頻繁に出力される正常な記録によって異常が埋没してしまい，本章で対象とする異常の発見が容易ではない．文献[62]においても，シングルサインオンでリソー

ス利用が可能なグリッドではログファイルから管理者が不正利用を検出することが困難であることが記されている．また，4つの異常のうち異常（A）と（B）は，ログファイルに出力される識別名とIPアドレスから直接発見することはできず，管理者はそれらがどの組織に管理されているかを調査しながらログファイルを閲覧する必要があり，このことも異常の発見を困難にする要因となっている．

さらに，このようなログファイルから異常の発見ができたとしても，その原因の分析に多大なる時間を要する場合がある．管理者は異常を発見すると，それが不正利用や障害によるものなのか，ユーザの誤操作などによる無害なものなのかを迅速に判断して対応する必要がある．原因の分析のためには，まずユーザ側に原因があるのか，またはサービス側やスケジューラに原因があるのか，問題の切り分けが行われる．この際，他組織でも同様の異常が発生しているかが重要な手がかりとなるが，他組織の管理者に電子メールや電話で問い合わせて異常の有無を確認することには，相手管理者の都合や時差によって時間を要する場合がある．特に，異常（B）はシングルサインオンのための権限委譲でも発生し，多くが正当なリソース要求であるため，すべてを調査することには多大な時間を要する．

上述した理由から，管理者がログファイルを調査して異常の発見，及び，原因の分析を行うことには多大な労力と時間を要することが問題となっている．この問題を解決するために，以下に示す管理者の4つの要求を満たし，統合的に管理者を支援するセキュリティモニタリングシステムの実現が課題となっている．

要求（R1）ログ情報からの異常の抽出

4.2.2節に記した4つの異常を抽出する機能が必要である．

要求（R2）IPアドレスと識別名の自動的な組織への対応付け

異常の抽出を行う際にIPアドレスと識別名を自動的に組織名に対応付け，組織単位での分析が行えることが必要である．

要求（R3）他組織の情報の許可された範囲内での参照

セキュリティモニタリングシステムを通して他組織のログ情報をすばやく参照できるようにする必要がある．同時に，アクセス制御を行い機密を保護する必要がある．

要求（R4）ユーザとクライアントの組織が異なる場合の効率的な分析

異常（B）の原因調査において，不正利用である可能性の高いものを抽出し，管理者が効率的に分析できるようにすることが必要である．

4.3 既存モニタリング手法

本節では，4.2節の要求機能の分析結果に基づき，グリッドで利用されている既存のモニタリング手法を調査する．本節では，まずグリッド以外でも利用されている既存の一般的なモニタリング手法の特徴を示し，4.2.2節で挙げた異常の発見に対しての有効性を調

表 4.1 既存手法と要求への対応.

	一般			グリッド	
	Snort, Bro	Logwatch	NVisionCC	SCMSWeb	MOGAS
(R1) 異常抽出	×ネットワーク	○	×クラスタ	×	×
(R2) 組織の対応付け	×	×	×	○	○
(R3) 他組織の情報	×	△	×	○	○
(R4) 効率的な分析	×	×	×	×	×

査する．次に，グリッドに特化して開発された既存のモニタリング手法の特徴を説明し，本研究で開発するセキュリティモニタリングシステムとの相違を示す．表 4.1 は本章における調査結果をまとめたものである．

4.3.1 一般的な既存モニタリング手法

グリッドを構成するそれぞれの組織内において，ネットワーク上の異常の発見にはネットワーク IDS（Intrusion Detection System：侵入検知システム）が利用される．Snort [63] と Bro [64] はオープンソースで開発されているネットワーク IDS であり，ネットワーク中のパケットを捕捉し，あらかじめ定義された攻撃パターンと比較することで侵入の兆候を検知する．ネットワーク IDS は Web サーバやメールサーバへの侵入を検知するために広く利用されている．要求（R1）ではログ情報から異常を抽出することが求められているが，グリッド認証基盤は暗号を使用して認証を行うため，パケット捕捉によって認証，認可の結果などログ情報に相当する情報を収集することは不可能である．仮に収集できたとしても，要求（R2），(R3)，(R4）では組織単位での分析が求められており，このような複雑な分析を攻撃パターンとして定義することは困難である．

単一コンピュータの異常発見にはホスト IDS が利用される．Logwatch [65] はオープンソースで開発されているホスト IDS であり，ログファイルをモニタリングし定期的にレポートファイルを作成する．正規表現を用いた文字列の比較や，比較によってマッチした回数を計測する機能があり，要求（R1）において異常（C）と（D）の抽出が可能である．また，ネットワークを介してログファイルを転送するソフトウェアと組み合わせることで，要求（R3）を満たすことも可能であると考えられる．しかし，組織に関する異常（A）と（B）を単純な正規表現で表現することはできず，要求（R1）を完全に満たすことはできない．また，組織単位での分析を行うためには正規表現よりも柔軟性の高い言語が必要であり，要求（R2）と（R4）を満たすこともできない．Tripwire [66] はオープンソースで開発されているホスト IDS であり，管理者の意図しない重要なファイルの書き換えを検

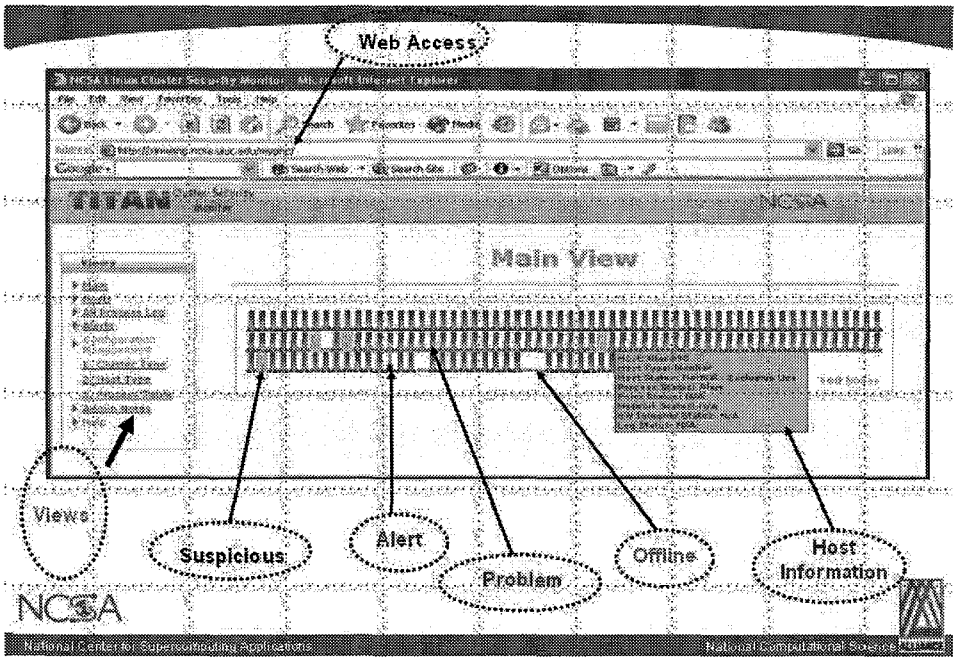

図 4.2 NVisionCC の可視化画面．文献 [67] より引用．

知する．書き換えの検知のためには，過去に算出したファイルのハッシュ値と現在のハッシュ値を比較する方法をとっており，ファイルの内容には関知しない．そのため，ログ情報の分析は行えず，いずれの要求を満たすこともできない．

HPC クラスタに特化したセキュリティモニタには NVisionCC [67] がある．NVisionCC はクラスタを構成するリソースが同質であるという特性を利用し，計算と関係のないプロセスの検知や，リソース間のファイルの差異検知機能を実装している．NVisionCC はこれらの情報を画像や表で可視化し，Web インタフェース上の画像として示す（図 4.2）．これにより管理者は，想定されていないプロセスの実行やファイルの書き換えなどの異常を視覚的に確認することができる．NVisionCC は組織間でのクラスタ共有は想定しておらず，組織外からのリソース要求における認証，認可のログ情報は扱わない．そのため，いずれの要求を満たすこともできない．

4.3.2 グリッドに特化した既存モニタリング手法

多くのグリッドではリソースモニタが導入されている．リソースモニタは CPU やメモリの使用率，コンピュータの稼働率などを可視化し，グリッドの利用状況を管理者とユーザに示すものである．代表的なグリッドのためのリソースモニタに SCMSWeb [39] がある．SCMSWeb はクラスタを構成するコンピュータからリソース情報を階層的に収集し，グリッドのような大規模な環境でも一元的にリソース情報を Web インタフェース上に可

視化することができる．この情報は，組織ごとに分類されて可視化されるため要求（R2）を満たしており，また，グリッドを構成する組織のすべてのユーザと管理者によって共有できるため要求（R3）も満たしている．しかし，リソースモニタは認証，認可のログ情報を扱わないため，要求（R1）と（R4）を満たしていない．このため，不正利用や障害によって想定外の負荷がかかっていることなどは検出することはできるが，4.2.2 節に記した 4 つの異常を抽出することはできない．

近年，グリッド技術のビジネス応用が盛んに行われるようになっており，アカウンティングシステムを導入したグリッドが増加している [68]．アカウンティングシステムは課金などの目的のために，ユーザがどれだけリソースを利用したかを計測する．Globus Toolkit を用いて構築されたグリッドを対象としたアカウンティングシステムである Multi-Organisation Grid Accounting System（MOGAS）[69, 70] は，各リソースのジョブ管理機構からジョブ情報を収集し，中央データベースに蓄積する．中央データベースにはユーザと組織の対応関係も保存されており，MOGAS は蓄積されたジョブ情報とこの対応関係の両方を用いて，組織ごとのリソース利用量を Web インタフェースに表示する．管理者とユーザはこれを閲覧することで組織ごとに分類されたリソース使用量を知ることができ，要求（R2）と（R3）を満たしている．しかし，MOGAS はジョブ管理機構から情報を収集するのみであり，認証，認可のログ情報からは情報を収集しないため要求（R1）と（R4）は満たしていない．このため，4.2.2 節で挙げた異常の発見を行うことはできない．このように，グリッドを対象としたリソースモニタとアカウンティングシステムは既に実用化されているが，認証，認可のログ情報を扱うセキュリティモニタリングに関しては研究がほとんどなされおらず，実用化されていないのが実情である．

4.4 提案手法と実装

本研究では，4.2.3 節で挙げた 4 つの要求を満足するセキュリティモニタリングシステムを提案する．以下では，提案システムとその特徴を述べ，複数組織に散在する情報の収集と蓄積，及び，異常発見の手法を説明する．

4.4.1 提案手法の概要

提案システムは，グリッド特有の複数組織にまたがる不正利用と関連しうる異常の発見とその原因分析にかかる管理者の負担を低減するために，複数組織のリソースに散在する数万行のグリッド認証基盤のログファイルに出力される認証と認可の情報を収集，蓄積，可視化し，管理者らが視覚的に異常発見と原因分析を行うことを可能とする．複数組織にまたがる各リソース上でのログファイルからの情報取得は，本システムの導入にかかる管

理者の負担を小さくするために，オペレーティングシステムに依存せずコンパイルが不要なスクリプトで実現する．情報の送信にはHTTPSのPOSTメソッドを使用し，安全性と互換性を両立する．取得した情報は各リソースから中央データベースに送信して集中的に管理することで，運用にかかる組織の管理者の負担が増大しないよう配慮する．蓄積された情報を総合的に解析し，複数組織にまたがる事象を管理者が容易に発見できるよう，グラフと表を用いて異常を抽出する手法をWebインタフェース上で実現する．このWebインターフェースには管理ポリシの異なる複数の組織の管理者が安全に情報を共有するためのアクセス制御機構を組み込む．

提案システムの構成を図4.3に示す．本システムでは複数組織に分散したリソースから情報を収集して分析する．このため，本システムは既存のアカウンティングシステムであるMOGASと同様に，各リソースから情報を収集して中央データベースに格納する．MOGASはジョブ管理機構からジョブ情報を収集するのに対し，本システムはグリッド認証基盤からセキュリティ情報を収集する．

図4.3中（a）の情報取得機構であるセキュリティセンサは，取得した情報からデータベースを更新するためのファイルを生成する．このファイルは図4.3中（b）のアップローダによって図4.3中（d）のWebインタフェースに安全に送信される．Webインタフェースはこのファイルを受信すると，図4.3中（c）のデータベースを更新する．このように，各リソースから得られた情報はデータベースに蓄積される．管理者が異常発見を行うためには，あらかじめMyProxyに管理者のプロキシ証明書を保存しておき，Webインタフェースにユーザ ID とパスワードを入力してログインする．WebインタフェースはMyProxyからプロキシ証明書を取り出して管理者を認証し，データベースに蓄積されている情報を用いて異常抽出と原因分析の機能を提供する．

4.4.2 複数組織にまたがる安全な情報取得

4.2.2節に記した4つの異常を発見するためには，リソースを要求したユーザの情報と，認証，認可の結果を複数組織から取得することが必要である．Globus Toolkitを用いて構築されたグリッドでは，グリッド認証基盤を提供するGSIの上で各サービスが提供される．計算リソースへの要求を受け付けるサービスはGRAMである．GRAMサーバではgatekeeperとjobmanagerと呼ばれる2つのプログラムが連携して要求管理を行う．クライアントからの接続を受け付けるプログラムはgatekeeperであり認証と認可もここで行われる．gatekeeperはリソースが要求された際にGSIの機能を呼び出して`globus-gatekeeper.log`という名前のログファイルに時刻，クライアントのIPアドレス，認証や認可の結果，エラーコード等を記録する．ログファイルの例を図4.4に示す．これは，ユーザ“Shingo Takeda”がGRAMサーバに接続し，正常に認証と認可が行

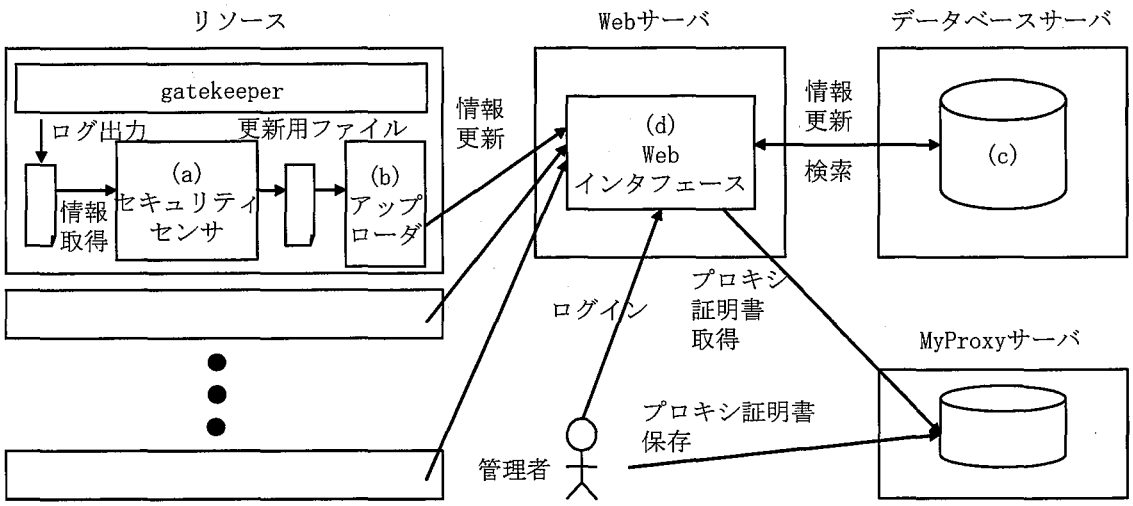

図 4.3 提案するセキュリティモニタリングシステムの構成.

われ，ローカルスケジューラの SGE に対するジョブが受け付けられた際の出力である．認証と認可に成功すると，gatekeeper は jobmanager に処理を渡し，jobmanager は指定された方法で計算プログラムを実行する．jobmanager はログファイルに情報を記録しない．このような処理の流れのため，GRAM では `globus-gatekeeper.log` からリソースを要求したユーザの情報と，認証，認可の結果を取得することが可能である．このため，本システムはこのログファイルを周期的に読み取り，追記があればそれを解析してデータベース更新用ファイルを生成する．

図 4.3 の (a) に示すセキュリティセンサは，リソース上で異常の発見と分析に必要な情報を取得し，データベースを更新するためのファイルを生成する役割を持つ．コンパイラやライブラリの追加インストールを必要とする実装では，セキュリティセンサを配置するために組織の管理者の負担が増大する．そこで，本システムのセキュリティセンサは Perl で実現することで負担が増大しないようにする．Perl は特殊な用途のものを除いてすべての Linux ディストリビューションが標準で対応しており，コンパイルも不要なため展開が容易である．また，本セキュリティセンサは 1,000 行未満の単純なプログラムで実現し，組織の環境に合わせた変更やログファイルの仕様変更にも対応が容易である．更新用ファイルに含まれる情報は，時刻 (時差を考慮して UTC で統一)，ユーザとサービスの識別名，クライアントとサーバの IP アドレス，認証の成否，認可の成否，GRAM のエラーコード，マッピングされたオペレーティングシステム上でのユーザのユーザ名，ユーザ ID，グループ名，グループ ID である．セキュリティセンサがログファイルの追記を想定外の文字列の出現などで解析できなかった場合，その文字列を更新用ファイルに出力する．更新用ファイルの例を図 4.5 に示す．この例では 2 つのジョブ要求が記録されてお

```
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 6: globus-gatekeeper pid=26649 starting at
 Thu Jul  6 00:31:30 2006
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 6: Got connection 127.0.0.1 at Thu Jul
6 00:31:30 2006
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 5: Authenticated globus user:
 /O=JP/OU=Osaka University/OU=Cybermedia Center/OU=hpc.cmc.
osaka-u.ac.jp/CN=Shingo Takeda
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 0: GRID_SECURITY_HTTP_BODY_FD=6
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 5: Requested service: jobmanager-sge
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 5: Authorized as local user: shingo
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 5: Authorized as local uid: 500
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 5:            and local gid: 10
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 0: executing /usr/local/globus/libexec
/globus-job-manager
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 0: GRID_SECURITY_CONTEXT_FD=9
TIME: Thu Jul  6 00:31:30 2006
 PID: 26649 -- Notice: 0: Child 26650 started
```

図 4.4 GRAM サーバのログファイルの一部.

り，前者は GRAM サーバとの接続テスト（ping）を行ったもので認証と認可ともに成功している．後者は SGE のジョブ実行を要求したもので，これについても認証と認可は共に成功している．

4.4.3 安全かつ管理の容易な情報収集

更新用ファイルの転送には，リソースからデータベースサーバに転送する PUSH 型と，逆に，データベースサーバがリソースから取得する PULL 型が考えられる．PULL 型の転送では各組織のリソースで外から内への通信をファイアウォールで許可する設定が必要となり管理者の負担が増大する．また，ログファイルへの追記の有無を調べるために毎回通信が必要となるため非効率である．このような理由から，本システムでは PUSH 型を採用し，リソースからデータベースサーバの方向に更新用ファイルを転送する．図 4.3 の (b) に示すアップローダは，セキュリティセンサが生成するデータベース更新用ファイルをデータベースサーバに転送する役割を持つ．アップローダもセキュリティセンサと同様に，各リソースに配置するため，コンパイラやライブラリの追加インストールを必要とする実装では組織の管理者の負担が増大する．本システムではアップローダをシェルスクリプトで実現し，配置と組織の環境にあわせた変更を容易にする．

```
INSERT IGNORE INTO globus_gatekeeper_session VALUES (
'/O=JP/OU=Osaka University/OU=Cybermedia Center/CN=host
/cafe01.exp-net.osaka-u.ac.jp',
'133.1.69.141',
'2007-01-05 19:49:59',
'2007-01-06 04:49:23' - INTERVAL 09 HOUR,
'2006-08-24 05:00:00',
'17533',
'198.202.74.232',
'/C=US/O=SDSC/OU=SDSC/CN=Ichiro Suzuki/UID=ichiro',
'S', NULL, NULL, NULL, 'S',
'ichiro', 'ichiro', '507', '507',
'jobmanager [PING ONLY]', '(ping success)', NULL);
INSERT IGNORE INTO globus_gatekeeper_session VALUES (
'/O=JP/OU=Osaka University/OU=Cybermedia Center/CN=host
/cafe01.exp-net.osaka-u.ac.jp',
'133.1.69.141',
'2006-11-07 06:32:00',
'2006-11-07 15:31:58' - INTERVAL 09 HOUR,
'2006-08-24 05:00:00',
'31950',
'163.220.2.59',
'/C=JP/O=AIST/OU=GRID/CN=Taro Yamada',
'S', NULL, NULL, NULL, 'S',
'yamada', 'yamada', '520', '520',
'jobmanager-sge', '/usr/local/globus/libexec
/globus-job-manager', '31951');
```

図 4.5 データベース更新用ファイルの例.

シェルスクリプトから起動するファイル転送コマンドには，多くのリソース上で利用できる汎用的なものを使用する必要がある．File Transfer Protocol（FTP）は代表的なファイル転送プロトコルであり，FTP クライアントの機能を持ったファイル転送コマンドは多くの環境で利用できる．しかし，公開鍵認証や転送データを暗号化する機能を備えたものは少なく，安全性の確保が難しい．このため，多くの管理者は安全にファイルを転送するために Secure Shell（SSH）の scp コマンドを使用している．scp では公開鍵認証と転送データの暗号化によって安全にファイルを転送できるが，サーバ側にシェルアカウントが必要であるため，リソースの数だけシェルアカウントを作成することはデータベースサーバの管理を困難にする．Hypertext Transfer Protocol（HTTP）は Web で利用されているプロトコルであり，HTTP クライアントの機能を持ったコマンドは多くの環境で利用できる．これらのコマンドは公開鍵認証と転送データの暗号化を行う HTTPS プロトコルに対応したものも多く，安全性の確保が可能である．しかし，ファイル転送のためのプロトコルではないため，Web サーバ上にファイルを受信する機能が必要となる．このような考察から，本システムでは Web インタフェースにファイルを受信する機能を持たせ，HTTPS でファイルの転送を行うことで，安全性と保守性を確保する．セキュリティセンサによってデータベース更新用ファイルが生成されると，アップローダはこのファイルを図 4.3 中 (c) の Web インタフェースに送信する．転送には多くの Linux ディストリビューション

```
#!/bin/sh
USER="cafe01"
PASS="97PDPNE3PBCO"
CACERT="/home/gridacct/mogas2.5.1/working/equifax.pem"
for i in $* ; do
 echo "https://$USER:$PASS@database.server.net/gwatch/PostSQL" \
 | wget  -i - \
 --post-file="$i" \ \
 --ca-certificate="$CACERT" \
 https://database.server.net/gwatch/PostSQL \
 && mv "$i" "$i.uploaded"
done
```

図 4.6 wget コマンドを使用するアップローダの記述例.

に付属している wget 等のコマンドで HTTPS による転送を行う．アップローダの記述例を図 4.6 に示す．この例では，wget コマンドでデータベースサーバに HTTP の POST メソッドを送信し，更新用ファイルをアップロードしている．接続先の URL は本システムの Web インタフェースで，Web インタフェースは送信された内容を元にデータベースを更新する．この通信ではサーバ認証を HTTPS で，クライアント認証を HTTP の BASIC 認証で行い，BASIC 認証のための ID とパスワードはリソースごとに与えられる．

4.4.4 異常発見と原因分析に向けたデータの蓄積

データベースの配置には，MOGAS のような一元管理，SCMSWeb のような階層管理，組織ごとの分散管理が考えられる．SCMSWeb ではクラスタのすべてのノードから情報を収集するためデータ量が多く，階層構造にすることで拡張性を高めているが，本システムで設置するセキュリティセンサは組織外からのアクセス要求を受け付ける GRAM サーバが対象であり，本システムのデータベースに求められる拡張性は比較的小さい．データベースの数が増加するとそれに伴って管理者の負担が増大するため，本システムでは，管理の容易さを重視し，取得した情報を中央データベースサーバで一元的に管理する．

図 4.3 の（c）に示すデータベースは，各リソースから収集されたログ情報，識別名と IP アドレスの組織との対応付け，及び，本システムのアカウント情報を格納する役割を持つ．これらの情報は構造が単純であり一般的なリレイショナルデータベース（関連データベース）に格納できるため，オープンソースで開発されている関連データベースで軽量性に定評のある MySQL を使用する．

ログ情報の保存にはデータベースの 2 つのテーブルを用いる．1 つは表 4.2 に示す globus_gatekeeper_session で，ここにはセキュリティセンサで解析できた情報が記録される．複数のリソースから得られたログ情報を分析するためには，各リソースでの

表 4.2 テーブル globus_gatekeeper_session の項目．下線は主キー．

項目名	型	用途
service_dn	文字列	要求先サービスの識別名
server_ip	文字列	要求先サーバの IP アドレス
db_time	日付	データベースに記録された時刻
log_time	日付	ログファイルに記載されている時刻
logmon_time	日付	セキュリティセンサが読み取った時刻
pid	整数	gatekeeper のプロセス ID
client_ip	文字列	要求元クライアントの IP アドレス
user_dn	文字列	要求元ユーザの識別名
authn_result	文字列	認証結果（S が成功，F が失敗）
authn_error_major	文字列	認証エラーコード（上位ビット）
authn_error_minor	文字列	認証エラーコード（下位ビット）
authn_error_token	文字列	認証エラートークン
authz_result	文字列	認可結果（S が成功，F が失敗）
user_local_name	文字列	OS 上でのユーザのユーザ名
user_local_group	文字列	OS 上でのユーザのグループ名
user_local_uid	整数	OS 上でのユーザのユーザ ID
user_local_gid	整数	OS 上でのユーザのグループ ID
requested_service	文字列	要求されたローカルスケジューラ名
invoked_program	文字列	実行されたプログラムのファイルパス
program_pid	整数	実行されたプログラムのプロセス ID

時刻が同期していることが重要である．世界中に組織が分散するグリッドではリソースによって時差があるため，本システムでは時刻を UTC で統一し，リソースの時計に異常が合った場合を考慮してデータベースに記録された時刻，ログファイルに記載されている時刻，セキュリティセンサが読み取った時刻の 3 つの時刻を保存しておく．もう 1 つのテーブルは表 4.3 に示す globus_gatekeeper_unknown で，ここにはセキュリティセンサで解析ができなかった文字列が記録される．

識別名と IP アドレスの組織との対応付けには 3 つのテーブルを用いる．ユーザの識別名と組織の対応は，表 4.4 に示す grid_user_organization に保存する．組織コードは組織ごとに一意に割り当てられた文字列である．同様に，サービスの識別名と組織の対応は，表 4.5 に示す grid_service_organization に保存する．IP アドレスと組織

表 4.3 テーブル globus_gatekeeper_unknown の項目．下線は主キー．

項目名	型	用途
service_dn	文字列	要求先サービスの識別名
server_ip	文字列	要求先サーバの IP アドレス
db_time	日付	データベースに記録された時刻
logmon_time	日付	セキュリティセンサが読み取った時刻
line_number	数値	行番号
line_string	文字列	ログファイルに記載されている文字列

表 4.4 テーブル grid_user_organization の項目．下線は主キー．

項目名	型	用途
user_dn	文字列	ユーザの識別名
organization	文字列	組織コード

表 4.5 テーブル grid_service_organization の項目．下線は主キー．

項目名	型	用途
service_dn	文字列	サービスの識別名
organization	文字列	組織コード

表 4.6 テーブル grid_host_organization の項目．下線は主キー．

項目名	型	用途
ip_range	文字列	IP アドレスの範囲
organization	文字列	組織コード

の対応は，表 4.6 に示す grid_host_organization に保存する．ip_range には単一の IP アドレスを登録できるほか，IPv4 アドレスの範囲を指定して組織に割り当てられている IP アドレスをまとめて登録することも可能である．

本システムのアカウントは表 4.7 に示す account に保存する．このアカウント情報を用いた認証と認可に関しては 4.4.5 節で説明する．

表4.7 テーブル account の項目．下線は主キー．

項目名	型	用途
user_name	文字列	アカウント名
user_pass	文字列	パスワード（テスト用で通常は使用しない）
user_dn	文字列	ユーザの識別名
role	文字列	RBAC 用ロール
expire	時刻	有効期限

4.4.5 複数管理者によるモニタリングが可能なインタフェース

本システムは，4.2.1 節で述べたようにグリッド管理者がグリッドシステム全体を主導し，組織の管理者らが連携してグリッドを運用するという形態を想定している．このため，モニタリング機能を提供するインタフェースには複数の管理者が同時に利用できることが求められる．本研究では，地理的に分散した管理者らが異常発見と原因分析を行えるように，これらの機能を図 4.3 中（d）の Web インタフェースで提供する．

本インタフェースでは管理者を GSI で認証する．Web での GSI 認証の実現のためには，多くのグリッドポータルと同様に MyProxy の証明書レポジトリを使用する．管理者がログインするには，MyProxy サーバにプロキシ証明書を登録しておく必要がある．管理者のためのログイン画面を図 4.7 に示す．ここで管理者はプロキシ証明書を登録した MyProxy サーバを選択し，アカウント名と MyProxy サーバで設定したパスワードを入力する．本インタフェースは選択された MyProxy サーバからプロキシ証明書を取得し，検証する．

本インタフェースは，管理者が異常の発見と分析，及び，本システムの管理を行うための機能を提供する．図 4.8 は管理者による異常発見と原因分析の流れを示したものである．本インタフェースの設計と実装はこのフローチャートに基づいている．4.2 節で述べた 4 つの異常のうち，異常（A）と（B）の抽出のためには，ユーザ，クライアント，サーバ，サービス，及び，組織の組み合わせに着目した，リソース要求のグラフ化を行う．この機能については 4.4.6 節で詳細を述べる．異常（C）と（D）の抽出のためには，期間ごとのリソース要求回数に着目した表とチャートによる異常の抽出を行う．この機能については 4.4.7 節で詳細を述べる．管理者がログインすると，最初に異常（A）と（B）を抽出した画面が表示される．異常が発見されなければ，画面を切り替えて異常（C）と（D）を抽出した画面に切り替える．どちらかにおいて異常が発見されれば，IP アドレスまたは

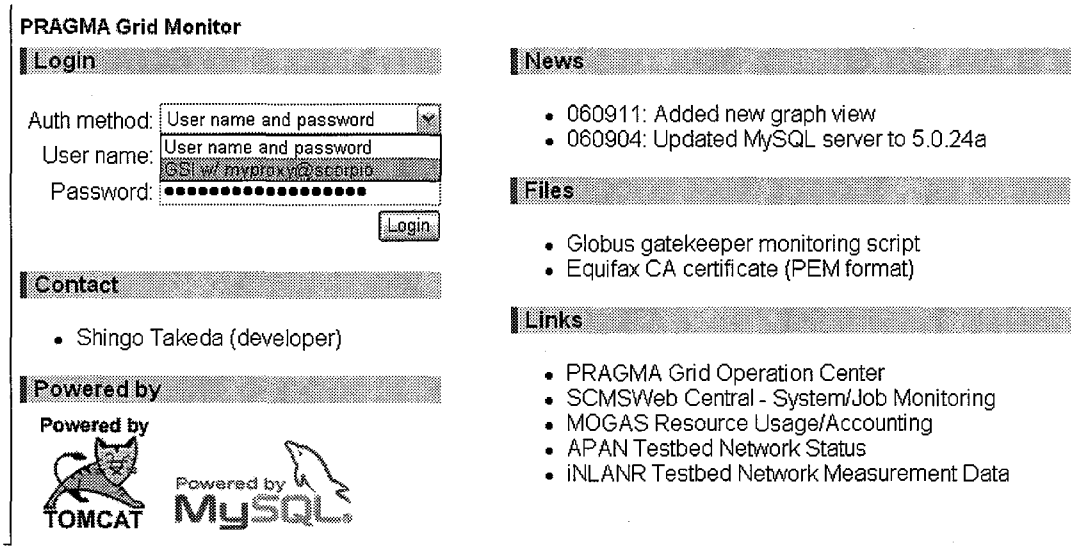

図 4.7 管理者のためのログイン画面.

表 4.8 アカウントに割り当てられるロールとその権限.

ロール名	対象	本システムの管理	異常発見	異常分析
superadmin	グリッド管理者	可	可	可
siteadmin	組織の管理者	不可	可	可
user	一般ユーザ	不可	自分のみ	不可

識別名を検索し，原因となっているユーザや組織を推定する．そして，必要に応じて電子メールや電話などの手段で原因を確認する．データベースへの対応付けの登録漏れが原因であった場合は管理画面で登録を行う．

本システムで扱うログ情報は機密性が高いため，本インタフェースでは GSI による認証を行ったうえで Role-Based Access Control（RBAC）[71] によるアクセス制御を行い，ログ情報の機密保護を行う．本システムでは図 4.1 に示した運用形態を想定し，表 4.8 に示すように，superadmin，siteadmin，user の 3 つのロールを考える．superadmin はモニタリングシステムの管理を含めてすべての操作が可能なロールで，グリッド管理者に割り当てることを想定している．siteadmin は組織の管理者で，異常発見と原因分析を行える．user は一般ユーザで，自分の情報だけを閲覧することができ，自分の証明書が他人に利用されていないことのみを確認できる．

ロールなど，アカウントの設定は管理画面で行う．管理画面の例を図 4.9 に示す．上部は本インタフェースのユーザ（グリッド管理者や組織の管理者）を追加するフォームであ

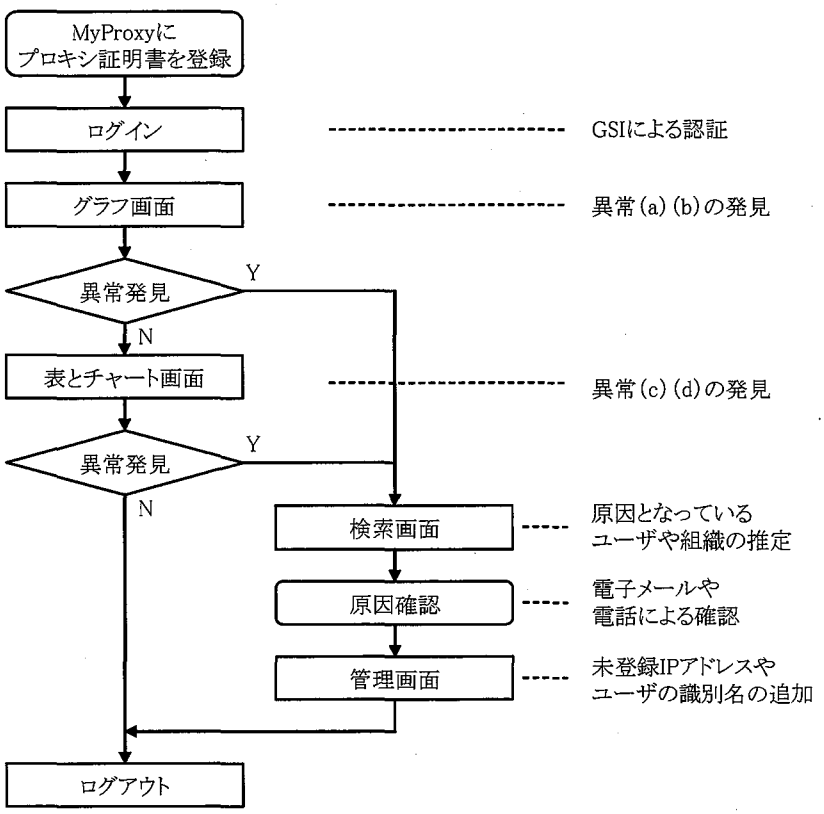

図 4.8 管理者の異常発見と原因分析における作業の流れ.

り，組織名，ロール，ユーザ名，パスワード，有効期限を設定することができる．下部は組織を追加するフォームで，管理コード，名称，国を設定することができる．これらの管理機能はグリッド管理者である superadmin のみが使用可能であるが，識別名や IP アドレスと組織を対応付けるための機能は，管理者の所属する組織に限って組織の管理者も使用できる．このように複数の管理者で対応情報を編集できるようにすることで，グリッド管理者に負担が集中し，対応が遅れないように配慮する．

4.4.6 グラフを用いた組織に関する異常の抽出

本節では，4.2 節で挙げた 4 つの異常のうち，異常（A）と（B）を抽出するための手法について述べる．シングルサインオンが実現されたグリッドにおいては，グリッド認証基盤の権限委譲の機能によって連鎖的にリソース要求が発生する．例えば，ユーザがグリッドポータルを介して計算リソースを利用する場合，ユーザとグリッドポータルの間で第 1 のリソース要求が，グリッドポータルと計算リソースの間で第 2 のリソース要求が発生する．このような環境において，異常（A）と（B）を抽出するためには，ユーザ，クライアント，サーバ，サービスがそれぞれどの組織に所属しているか，及びグリッド上で発生

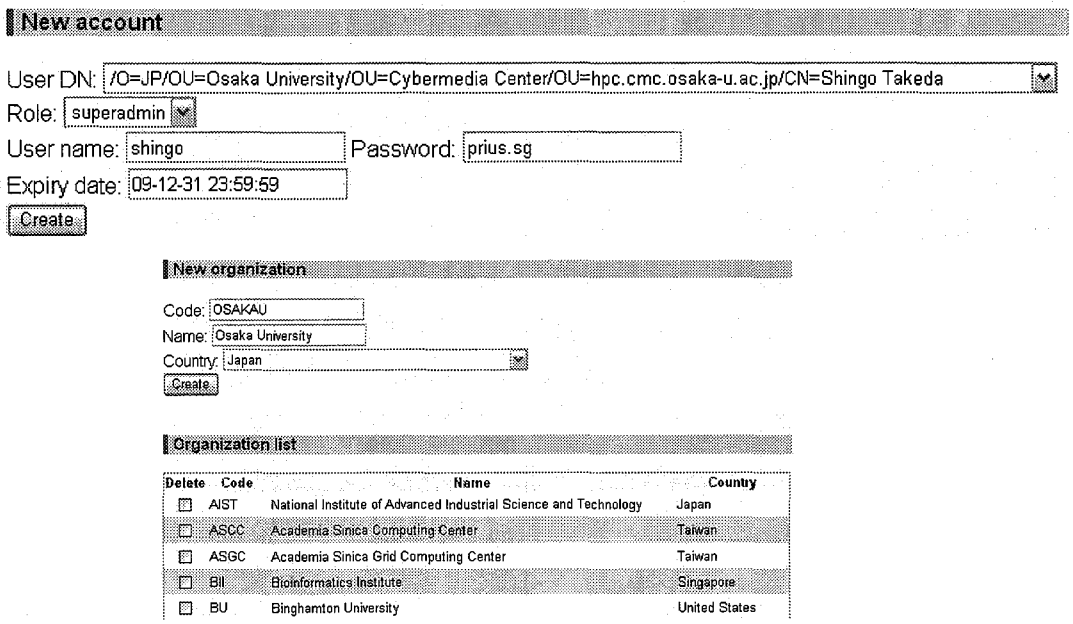

図 4.9 Web インタフェースの管理画面の例.

するこのような認証連鎖の把握が重要となる．本研究では，ユーザ，クライアント，サーバ，サービスの順に伝搬するリソース要求の流れをリソース要求フローと定義し，このリソース要求の最小単位となるリソース要求フローをグラフ化してログ情報を可視化することで，異常（A）と（B）を抽出する．これらの異常の効率的な抽出のためには，以下の 6 つの情報項目を管理者に分かりやすく示すことが重要である．

- 要求元ユーザ（ユーザ証明書の識別名）
- 要求元クライアント（IP アドレス）
- 要求先サーバ（IP アドレス）
- 要求先サービス（ホスト証明書の識別名）
- 上記 4 つの帰属する組織
- 要求の成否

これらの項目をグラフ化することで，管理者はどの組織のどのユーザが，どの組織のどのサービスを要求しているかをすぐに把握できる．また，その要求が成功したか，あるいは認証か認可が失敗したかを同時に示すことで，異常（C）と（D）がどのリソース要求フローで発生しているかを知ることができる．帰属する組織についてはデータベースを参照し，識別名か IP アドレスの少なくともどちらか一方が登録されなかった場合は，異常（A）として表示する．

本インタフェースでは図4.10に示すようにリソース要求フローを描く．ユーザ，クライアント，サーバ，サービスを水平方向に分類し，組織を縦方向に分類して，リソース要求フローを右から左へ描く．直線を使用すると複数の線が交わったときに区別がつかないため，スプライン曲線を使用する．成功した要求には緑色，失敗には赤色を用いて要求の成否を色で表現する．成功と失敗が同じリソース要求フローで両方発生していた場合は，異常発見のために重要性の高い失敗を優先し赤色を用いる．図4.10中（a）の緑線は組織ABCのユーザ“Taro Yamada”が同組織のクライアント“11.12.13.15”から組織XYZのサーバ“21.22.23.24”の上のサービス“service.xyz.net”に要求を送り，成功したものである．図4.10中（b）の赤線は組織XYZのクライアント“21.22.23.25”から同組織のサーバ“21.22.23.24”の上のサービス“service.xyz.net”に要求を送ったが，失敗したものである．これは認証エラーであったため，ユーザの識別子が記録されていない．

正常時は，ユーザとクライアント，サービスとサーバはそれぞれ同じ組織に属しており，リソース要求は成功する．管理者はこれを閲覧することで，リソースが正常に利用されていることを視覚的に確認できる．データベースに対応付けが保存されていない識別名かIPアドレスが要求に含まれていた場合は組織が不明となり，最下段のクエスチョンマークの領域に点が表示され，異常（A）を示す．管理者はグラフを見ることで，どのような流れの不明なアクセスが発生したかを即座に把握でき，また，類似したリソース要求フローを見ることで原因を推測することができる．異常（B）が発生すると，図4.11に示すようにクライアントとユーザの間で組織の境界をまたいだ線が描かれる．これは，ユーザ証明書の盗用によって発生している可能性があるが，多くの場合，GSIによる委譲やユーザの物理的な移動のために描かれたものであり，即座に問題であると判断することはできない．これをすべて管理者が調査することは非効率であるため，本システムではデータベースに蓄積されている過去のログ情報を使用し，最近に出現していないリソース要求フローを強調することで同じフローを何度も調査する手間を省く．本システムは，過去30日間の履歴を参照し，ユーザとクライアントの組み合わせが履歴になく，かつ，それらの組織が異なる場合には別の色の線を描くことで管理者に確認を求める．提案システムでは成功には水色，失敗には黄色を使用する．

4.4.7 表を用いた要求回数に関する異常の抽出

本節では，4.2節で挙げた4つの異常のうち，異常（C）と（D）を抽出するための手法について述べる．グリッド認証基盤のログファイルには大量の失敗が発生しても，それ以上に多くの成功が記録される場合が多く，管理者がテキスト形式のログファイルを見て異常を発見することは難しい．そこで本システムでは，一定時間内に発生した成功と失敗の回数に着目し，それぞれの回数を表で，成功と失敗の割合をチャートで示す．図4.12に，

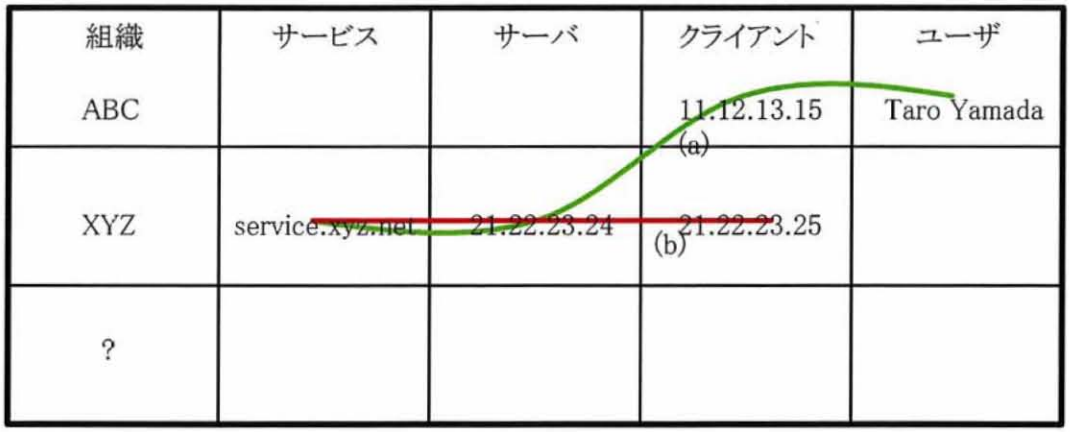

図 4.10 リソース要求フローのグラフ化.

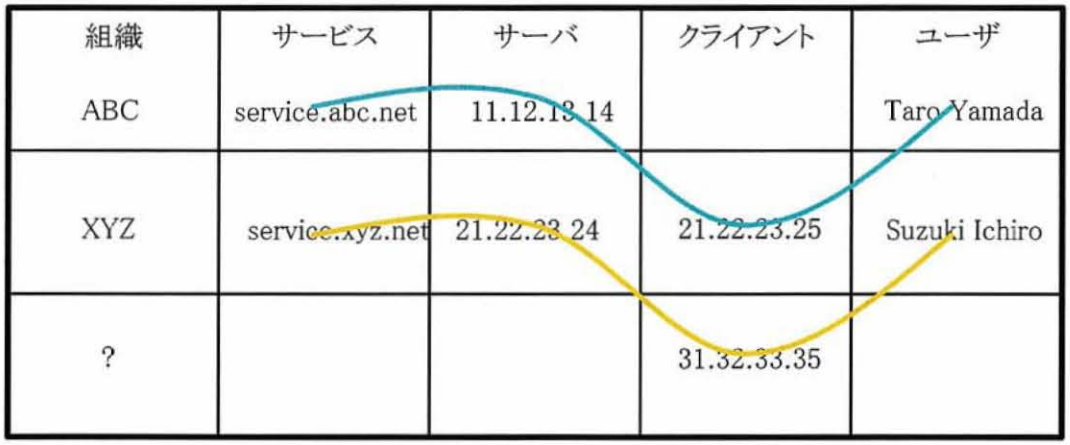

図 4.11 クライアントとユーザの組織が異なるリソース要求フローのグラフ化.

一定期間内の失敗回数を表で表示する例を示す．サービスとユーザの識別名は文字数が多くすべての表示が難しいため，CN（Common Name）の部分のみを表示し，マウスカーソルを合わせたときに識別名全体をポップアップ表示する．

異常（C）は数値の右にエクスクラメーションマークのアイコンを表示することで示す．図 4.12 中（a）はサービス “rs.osaka-u.ac.jp” で最近 1 時間以内に 42 回のリソース要求の失敗が発生したことを示しており，失敗回数が多いために異常（C）として抽出され，エクスクラメーションマークが表示される．同様に，しきい値を下回る成功回数にも，セキュリティセンサやアップローダに問題がある可能性が高いため，エクスクラメーションマークを表示する．警告を出すしきい値は Web インタフェースの設定ファイルで変更でき，グリッド規模に応じた値をグリッド管理者が設定可能である．異常（D）が発見された場合には，図 4.12 中（b）に示すようにサービスの識別名にエクスクラメーションマークを表示する．IP アドレスか識別名のリンクがクリックされた場合，それをキーとして

	hour	day	week	month	year
/CN=rs.osaka-u.ac.jp	(a) 42	160	165	172	191
/CN=trouble.osaka-u.ac.jp (b)	0	0	0	0	0
/CN=fumei.osaka-u.ac.jp (c)	0	3	12	54	180

図 4.12 表による一定期間内の失敗回数の表示.

データベースを検索し，それが含まれる履歴を表示して即座に異常分析に移行することができる.

4.5 グリッドテストベッドでの評価実験

本節では，本システムの有用性を評価するために，開発したセキュリティモニタリングシステムを Globus Toolkit で構築されたグリッドに展開し，実験を行う．まず，実験を行った環境と，セキュリティセンサとデータベースの配置について説明する．その後，実験結果を示し，考察を行う.

4.5.1 実験環境

実験は，PRAGMA（Pacific Rim Application and Grid Middleware Assembly）[72] のグリッドテストベッドで行った．PRAGMA はグリッドアプリケーションとミドルウェアの開発を行う研究コミュニティであり，主に環太平洋地域の 30 を超える大学と研究機関が参加している．PRAGMA ではリソースワーキンググループが中心となってテストベッドを構成する各組織の管理者を集め，グリッドの運用管理に携わる Grid Operation Center（GOC）を形成している．この GOC の代表者が本研究でのグリッド管理者に相当する．本テストベッドは Globus Toolkit を用いて構築されており，プログラムの遠隔実行のためには GRAM が使用される.

本システムの評価環境を図 4.13 に示す．セキュリティセンサは National Institute of Advanced Industrial Science and Technology（AIST），San Diego Supercomputer Center（SDSC），Monash University（MU），HCMC Institute of Information Technology（IOIT-HCM），National Center for Supercomputing Applications（NCSA），Academia Sinica Grid Computing Center（ASCC）の各組織のリソースに 1 つずつ，及び，Osaka University（OSAKAU）の 2 つのリソースの計 7 つのリソースに配置した．これらのリソースはすべ

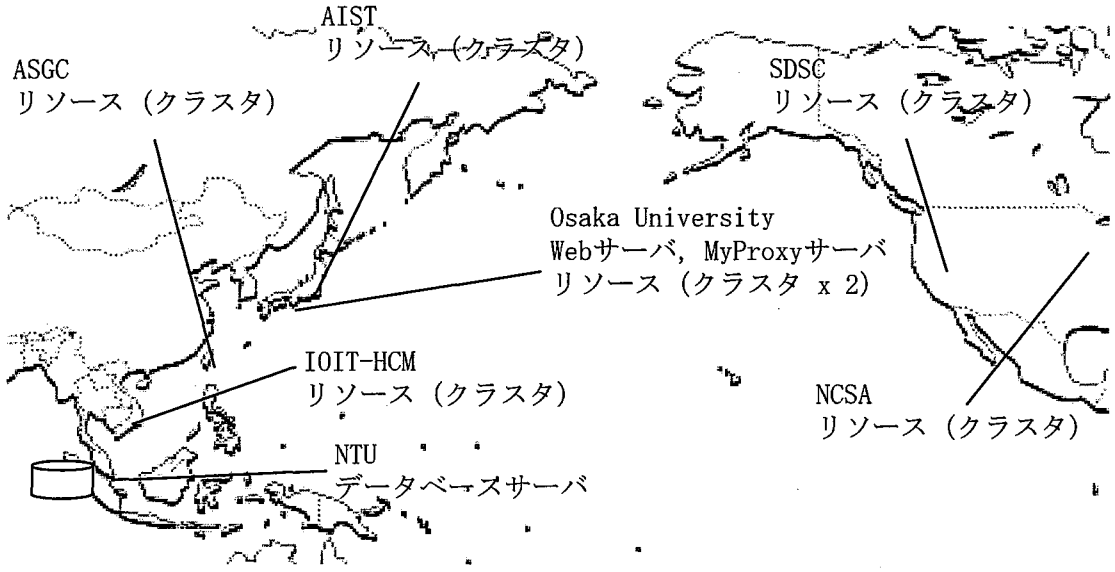

図 4.13 評価実験を行ったテストベッド構成.

て Linux で構築された HPC クラスタである．セキュリティセンサは組織外からの要求を受け付けるフロントエンドノードのみに配置し，ファイアウォール内の個々の計算ノードには配置していない．中央データベースは Nanyang Technological University (NTU) に配置した．Web インタフェースは Osaka University に配置し，中央データベースとは VPN で接続して実験を行った.

4.5.2 実験結果

2006 年 8 月 21 日から 2007 年 3 月 11 日までの 203 日間にセキュリティセンサでは 30 人のユーザからの成功 129,107 回（75%），失敗 43,043 回（25%），合計 172,150 回のリソース要求を検出した．失敗の内訳は認証に成功しなかったもの 42,542 回（99%），認証後に認可で失敗したもの 501 回（1%）であった．成功した要求の 86% 以上は上位 3 ユーザからのものであり，本テストベッド環境では多数の要求を出すユーザは限られていることがわかった．クライアントとユーザの組織の異なる要求は 9,211 回（7%）であった.

図 4.14 に期間中に表示されたグラフの例を示す．この例では異常は発生しておらず，すべてのリソース要求フローにおいてユーザとクライアントは同じ組織に属している．図 4.14 中（a）は組織 AIST のユーザ “Taro Yamada”（仮名）が同組織のクライアント “183.220.35.131” から組織 NCSA のサーバ “66.99.215.140” 上のサービス “tgc.trecc.org” にリソースを要求し，成功したものである．図 4.14 中（b）は組織 SDSC 内で認証エラーが発生していることを示しているが，異常であるか正常値の範囲内であるかは表とチャー

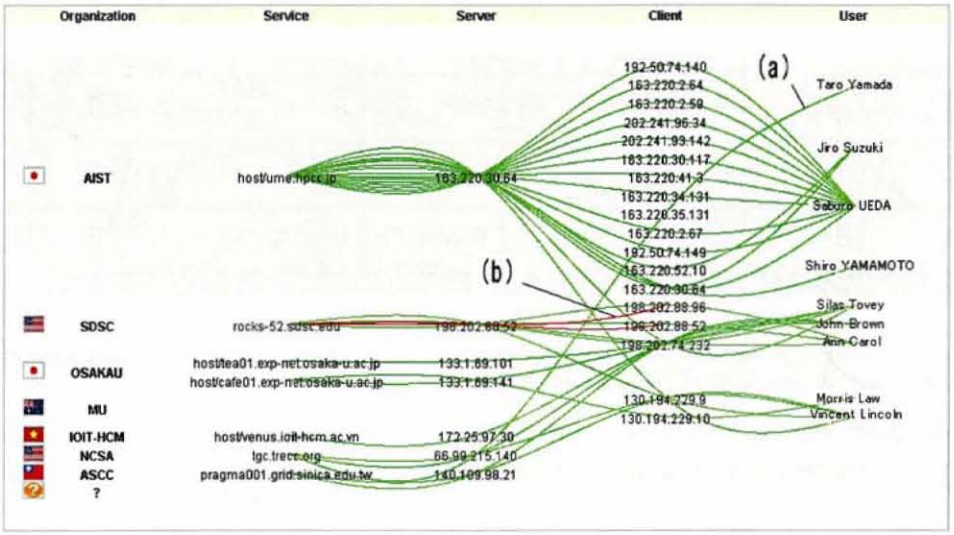

図 4.14 リソース要求フローが可視化されたグラフの例．表示期間は 1 か月．ユーザは仮名．

ト画面で調査する必要がある．

図 4.15 に期間中に表示された表とチャートの例を示す．図 4.15 中（a）は組織 AIST の “host/ume.hpcc.jp” で過去 1 日以内に 1,892 回のアクセス要求が成功したことを表している．図 4.15 中（b）は過去に成功したアクセス要求回数の組織ごとの割合を表している．図 4.15 中（c）は組織 SDSC の “rocks-52.sdsc.edu” で過去 1 か月以内に 8,894 回のアクセス要求が失敗したことを表している．この失敗回数は非常に多く，何らかの問題がある可能性が高いため，数値の右にエクスクラメーションマークのアイコンが警告として表示されている．

4.2 節で述べた 4 つの異常のうち，異常（A）は管理者がグラフを見ることによって発見することができた．データベースに組織との対応付けが未登録である識別名か IP アドレスを含んだ要求があった場合，不明な組織として最下段に異常（A）に該当するリソース要求フローが抽出された．抽出された異常には，ユーザ識別名のみが未登録である場合と，クライアント IP アドレスのみが未登録である場合の 2 種類があった．前者は，不正に生成されたユーザ証明書が使用されている可能性もあるが，新たなユーザ証明書が発行されたことが原因である可能性が高い．異常を発見した管理者が使用されたクライアントを管理する組織に問い合わせた結果，すべてが新たな証明書の発行が原因であったことが確認された．組織の管理者は，未登録であった識別名を組織に対応付けることで，この識別名を以後異常として検出されないようにできた．後者は，グリッドを構成する組織外の第三者にユーザ証明書が盗用されている可能性がある．この異常を発見した管理者が証明書の所有者であるユーザに問い合わせた結果，期間内に存在した原因は，ユーザの物理的な移動と，IP アドレスの登録漏れの 2 つであった．ユーザの物理的な移動は，グリッド技

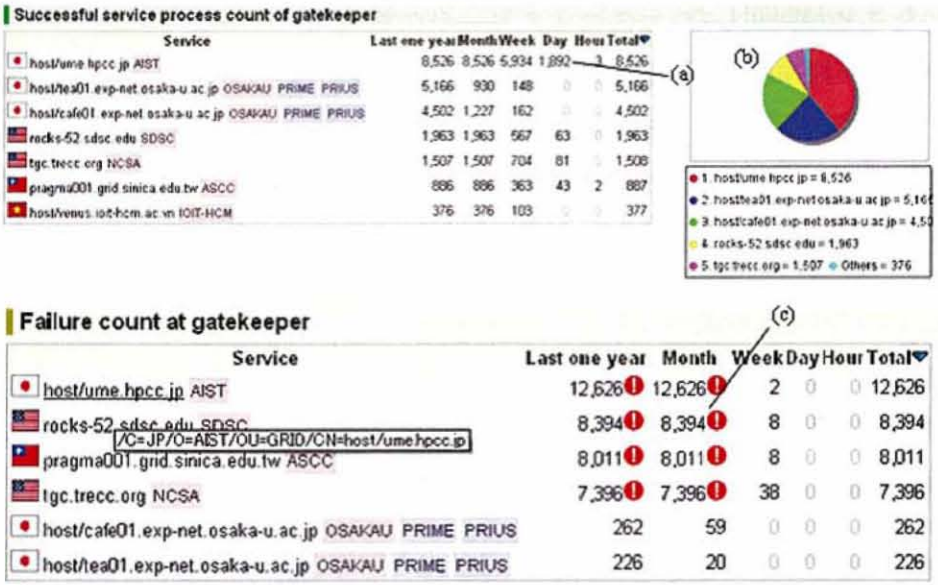

図 4.15 異常が抽出された表の例.

術のデモンストレーションなどのためにユーザが組織外のネットワークを一時的に使用していたもので，正当なリソース要求である．IP アドレスの登録漏れは，ユーザは組織内のネットワークを使用していたが，組織の管理者によって本システムに IP アドレスが登録されていなかったもので，IP アドレスを追加登録することで以後異常として検出されなくすることができた．

異常（B）の分析はグラフの履歴を用いた強調によって効率化された．本テストベッドでは，複数のグリッドポータルが使用されており，ユーザがグリッドポータルからリソースを利用すると，ユーザの権限がグリッドポータルに委譲され，クライアントがグリッドポータルとなる．このため異常（B）は日常的に頻繁に発生するため，すべてを調査することは困難であった．実験開始直後は，クライアントとユーザの組織が異なるリソース要求フローすべてが強調表示されたが，同じリソース要求フローが 30 日間に 2 度以上発生すると強調表示されなくなり，徐々に強調されるリソース要求フローの数は減少した．実験期間の後半に強調されたリソース要求フローを異常（A）と同様に管理者が調査したところ，グリッドポータルへの新規ユーザの追加，GSI による権限委譲，ユーザによる他組織のクライアントの借用が原因であることが判明した．

異常（C）は表に失敗回数と共に抽出され，管理者が発見することができた．しきい値を超える回数の認証失敗が発見された場合，異常を発見した管理者はクライアント IP アドレスをキーとしてデータベースを検索し，異常を発生させているユーザを推定することができた．管理者が原因と推定されたユーザに問い合わせたところ，誤ったユーザ証明書や期限の切れたプロキシ証明書を用い，シェルスクリプトによって繰り返してリソースを

要求していたことが判明した.

異常（D）は実験期間中に1度だけ発生し，表のサービスの識別名にエクスクラメーションマークが表示された．本システムのWebインタフェースでサービスの識別名をキーとして管理者が検索し，調査したところグリッド認証基盤のログファイルにファイルコピーの失敗メッセージが出力されていたことが分かった．リソース上でgatekeeperを調査したところ，ファイルのパーミッションの設定が誤っており，認証と認可に成功してもjobmanagerが正常に起動できない障害が発生していたことが判明した.

実験期間中に，悪意を持った攻撃であると見られる異常は発見されなかった.

4.5.3 考察

本セキュリティモニタリングシステムはグリッド管理者，及び，組織の管理者による異常発見と原因分析を容易にすることを目的に開発された．本節では，4.5.2節の実験結果をもとに，4.2.3節で述べた機能要件を満たしているかを考察する.

要求（R1）では，ログ情報から異常を抽出することが必要とされている．実験結果によると，実験環境では203日間に合計172,150回のリソース要求が発生した．これは，平均すると1日あたり848回の要求が発生したことを示している．GSIでは1回のリソース要求ごとにログファイルに30行程度が出力されるため，1日当たり約25,000行が出力されたことになる．このような大量のログ情報から異常を見落とさずに発見することは，管理者にとって大きな負担となっていた．本システムでは，異常をグラフや表の上に抽出することで，異常を視覚的に発見することを可能にした．リソース要求フローを可視化したグラフでは，サービス，サーバ，クライアント，ユーザの組み合わせの数だけ線が描かれ，図4.14の例では1か月の情報は26本の線となる．要求数に着目した表では，サービス（7つ）およびユーザ（30人）ごとに成功と失敗回数が要約され，蓄積されている全ての情報を74行で表示でき，しきい値を超える値は強調される．これらにより，本システムではログファイルを閲覧するよりも容易に異常の発見が可能である.

要求（R2）では，IPアドレスと識別名が自動的に組織へと対応付けられることが必要とされている．本システムは，中央データベースに対応関係を保存しておき，ログ情報から異常を抽出する際にこれを利用することで，自動的な対応付けを実現した．本システムでは，組織の管理者がそれぞれの組織の対応関係を編集することができ，すべての管理者で対応関係を共有することができる．実験期間中の172,150回のリソース要求に対して，自動的な対応付けによって異常（A）と異常（B）を抽出することができ，また，異常の原因となっている組織を推定することができることが確認された.

要求（R3）では，他組織のログ情報を，許可された範囲内での参照できることが必要とされている．本システムではグリッド管理者や組織の管理者は，他組織のサービスに向け

られたリソース要求フローや，他組織のサービス上でのエラー回数を Web インタフェース上で見ることができ，異常の分析において問題の切り分けのためにこれらの情報を活用することができる．

要求（R4）では，ユーザとクライアントの組織が異なる場合にも，効率的な分析ができることが必要とされている．本テストベッドではグリッドポータルが設置されており，ユーザとクライアントの組織は異なるが正当であるリソース要求は日常的に発生する．実験期間中に組織の異なる要求は 9,211 回で，1 日当たりの平均は約 45 回であり，これらの原因を全て調査することには大きな労力が伴う．これを効率化するため，本システムでは過去 30 日間に発生していない異常（B）に該当するリソース要求フローを抽出して強調する．これにより，同じリソース要求フローが 2 度以上発生することで強調される数が減少した．最終的に管理者が分析の対象とする要求の数は，ユーザの追加やリソースの構成変更の頻度に依存するが，本テストベッド実験では月に数個から数十程度に減少し，異常（B）の原因分析が効率化された．

以上から，本システムは 4.2.3 節で述べた機能要件を満たしていると言える．本評価実験を通じて，本システムを使用することによって，管理者が直接ログファイルを閲覧しなくともグリッドで発生する不正利用や機密漏えいに関連する可能性のある異常を発見できることも示された．

4.6 むすび

本章では，グリッド認証基盤からログ情報を収集し，不正利用や障害の前兆となる異常を管理者が視覚的に発見することを可能にするセキュリティモニタリングシステムを提案した．本章で提案したシステムは，グリッド特有の複数組織にまたがる不正利用と関連しうる異常の発見とその原因分析にかかる管理者の負担を低減するために，複数組織のリソースに散在する数万行のグリッド認証基盤のログファイルに出力される認証と認可の情報収集，蓄積，可視化し，管理者らが視覚的に異常発見と原因分析を行うことを可能とした．

複数組織にまたがる各リソース上でのログファイルからの情報取得は，本システムの導入にかかる管理者の負担を小さくするために，オペレーティングシステムに依存せずコンパイルが不要なスクリプトで実現した．情報の送信には HTTPS の POST メソッドを使用し，安全性と互換性を両立した．取得した情報は各リソースから中央データベースに送信して集中的に管理することで，運用にかかる組織の管理者の負担が増大しないよう配慮した．蓄積された情報を総合的に解析し，複数組織にまたがる事象を管理者が容易に発見できるよう，グラフと表を用いて異常を抽出する手法を Web インタフェース上に実現した．この Web インターフェースには管理ポリシの異なる複数の組織の管理者が安全に情報を

共有するためのアクセス制御機構を組み込んだ.

本システムはPRAGMAグリッドテストベッドに展開して評価実験を行った.評価実験によって,本システムが先に述べた要件を満たしており,管理者の負担を低減できることが示された.

今後の課題として,より汎用的で効率的なデータ管理方法の開発があげられる.現在の実装は,学術機関を結んだ非商用のグリッドテストベッドを対象として設計されている.ここではログ情報の集中管理が合理的であるが,商用のグリッドではログ情報を外部に出すことが組織のポリシに違反する可能性が高く集中管理は難しい.そのようなポリシを持つ組織には個別にデータベースを配置する必要がある.複数のデータベースを配置でき,ポリシに違反しないように他のデータベースと協調して問題を発見できるアーキテクチャへの移行が今後の課題である.

第 5 章

結論

5.1 本論文のまとめ

本論文では，グリッドでの科学技術計算において，ユーザによる位置透過的なプログラム展開を可能にするファイルアクセス手法と，管理者による視覚的な異常の発見と分析を可能にするセキュリティモニタリングシステムに関する研究開発の成果をまとめた．

1 章では，効率的な学際研究を支える情報技術として，シングルサインオンでリソース共有が可能なグリッドへの注目が高まる一方，安全性とシングルサインオンの利便性の両立が困難であることが，ユーザのグリッド利用と管理者のグリッド運用の障壁となっていることを述べた．そのような背景から，本研究ではユーザと管理者のそれぞれの視点で障壁を取り除くことがグリッドによる効率的な学際研究の実現に向けて重要な課題であることを述べ，その解決を本研究の目的とした．

2 章では，安全性と利便性の両立を困難としている原因を詳細に追及するために，グリッドで科学技術計算を行う際に利用される技術とその実装を調査した．その結果として，ユーザ視点からは，ユーザが自身で作成した計算プログラムをグリッドに展開してシングルサインオンで計算を行うためには，計算プログラムの拡張やファイルの配置に多大な労力を要することが問題となっていることを述べた．一方，管理者視点からは，シングルサインオンのために複雑かつ大量に発生する細分化されたリソース要求から，不正利用や機密漏えいの前兆となりうる異常を発見することが困難であることを述べた．さらに，これらの分析結果から，グリッド認証基盤に適応した安全かつ位置透過的なファイルアクセスと，グリッド認証基盤を安全性の観点からモニタリングするシステムの実現が，ユーザと管理者の負担を低減するのための急務であることを導いた．

3 章では，ユーザ視点からの課題であるグリッド認証基盤に適応した安全かつ位置透過的なファイルアクセス手法に関する研究開発の成果をまとめた．本研究では分散ファイルシステム SFS の高い位置透過性と互換性に着目し，SFS をグリッド認証基盤 GSI に適応

させることで，安全性と位置透過性を両立するファイルアクセス手法を提案した．提案手法では，SFS サーバに GSI-SFS 認証サーバを，SFS クライアントに GSI-SFS 認証クライアントをそれぞれ新たに実装して追加し，SFS を拡張した．グリッド認証基盤への適応のために，SFS では利便性を損なう要因の 1 つであった鍵の登録を，SFS の self-certifying pathname 検索機能と GSI の認証と暗号化の機能を応用して，安全かつ自動的に行うメカニズムを提案した．また，委譲証明書の制限のために，受け付けを許可または拒否するプロキシ証明書の識別名を GSI の機能を応用してサーバに送信し，ユーザが遠隔設定する機能の実現方法を提案した．本手法によって，ユーザはファイルの位置を意識することなくグリッドに既存の計算プログラムを安全に展開することが可能となった．提案手法を応用して，中国科学院の生物データを大阪大学の高性能クラスタシステムで解析するグリッドテストベッドを構築した．その結果，ユーザがグリッドポータルからシングルサインオンで既存の DNA 解析プログラムを利用できることを確認し，科学技術計算への有用性を示した．

4 章では，管理者視点からの課題であるグリッド認証基盤を安全性の観点からモニタリングするシステムに関する研究開発の成果をまとめた．提案システムは，グリッド特有の複数組織にまたがる不正利用と関連しうる異常の発見とその原因分析にかかる管理者の負担を低減するために，複数組織のリソースに散在する数万行のグリッド認証基盤のログファイルに出力される認証と認可の情報を収集，蓄積，可視化し，管理者らが視覚的に異常発見と原因分析を行うことを可能とした．複数組織にまたがる各リソース上でのログファイルからの情報取得は，本システムの導入にかかる管理者の負担を小さくするために，オペレーティングシステムに依存せずコンパイルが不要なスクリプトで実現した．情報の送信には HTTPS の POST メソッドを使用し，安全性と互換性を両立した．取得した情報は各リソースから中央データベースに送信して集中的に管理することで，運用にかかる組織の管理者の負担が増大しないよう配慮した．蓄積された情報を総合的に解析し，複数組織にまたがる事象を管理者が容易に発見できるよう，グラフと表を用いて異常を抽出する手法を Web インタフェース上に実現した．この Web インターフェースには管理ポリシの異なる複数の組織の管理者が安全に情報を共有するためのアクセス制御機構を組み込んだ．本システムによって，管理者がログファイルを詳細に調査しなくとも視覚的に異常を発見し，また発見された異常をデータベースに蓄積された履歴を用いて迅速に分析することが可能となった．本システムは主に環太平洋地域の大学や研究機関を結んだ PRAGMA グリッドテストベッドに展開し，評価実験を行った．実験の結果，管理者がテキスト形式のログファイルを調査して異常を発見することは困難である量のリソース要求が観測されたが，本システムを用いることで管理者は異常の発見と分析を行うことができ，本手法が管理者の負担を低減することを示した．

以上により，本研究ではシングルサインオンで計算リソースを共有するグリッドの構

築，運用にかかるユーザと管理者の負担を低減した．これにより，グリッド技術を用いた科学技術計算が容易となり，科学技術研究の効率化が期待される．

5.2 今後の課題

提案したファイルアクセス手法では，GSI と SFS の機能を利用して安全な認証とデータ転送を実現している．しかし，認可の機能に関しては GSI の単純なマッピング機能を使用しており，ファイル単位でのアクセス制御はオペレーティングシステムに依存している．このため，アクセス制御のためには `grid-mapfile` を参照してユーザの識別名とオペレーティングシステム上のユーザ名を対応づけながらアクセス権を設定しなければならない．そのため，特にユーザ数の多いグリッドでの利用には課題が残る．今後，OGF で行われているデータグリッドのアクセス制御に関する研究の動向を参考にしながら，ユーザの識別名を直接指定できるアクセス制御の機能を取り入れていくことが必要である．

提案したセキュリティモニタリングシステムは，自動的な侵入検知を行えるに至っていない．今後もグリッドのモニタリングを続けて情報を収集し，グリッドの特性をより明確にしていくことで統計的な手法の適用による侵入検知が可能になると考える．また，集中的なデータ管理では，機密保護の理由でログ情報を組織外に提供しない組織もあり，セキュリティセンサの配置ができないことがある．これについても，データグリッド分野で研究されているアクセス制御技術の成果を取り入れ，ログ情報の柔軟なアクセス制御を実現するなどの対応が求められる．

謝辞

本研究の全過程を通じて，懇切なる御指導，御助言と格別なる御配慮を賜りました大阪大学サイバーメディアセンター 下條真司教授に謹んで感謝の意を表します．

本論文をまとめるにあたり，貴重な時間を割いて頂き，懇切なる御指導と有益な御助言を賜りました大阪大学大学院情報科学研究科マルチメディア工学専攻 藤原融教授，薦田憲久教授，西尾章治郎教授，岸野文郎教授に心より感謝の意を表します．

本研究を推進するにあたり，直接の御指導，御助言，御討論を頂きました大阪大学大学院情報科学研究科 伊達進特任准教授に深く感謝いたします．

本論文を執筆するにあたり，多大なる御指導，御助言を頂きました大阪大学大学院情報科学研究科マルチメディア工学専攻 馬場健一准教授に深く感謝いたします．

本研究を進めるにあたり，多大なる御支援を頂きましたバイオグリッドプロジェクトの方々に深く感謝いたします．

セキュリティモニタリングシステムの研究開発において，御指導，御助言，御討論を頂きました Nanyang Technological University の Bu-Sung Lee, Francis 教授，Junwei Zhang 氏に深く感謝いたします．

セキュリティモニタリングシステムの評価実験において，御協力を頂きました Pacific Rim Application and Grid Middleware Assembly (PRAGMA) のメンバーの方々に深く感謝いたします．

本研究を進めるにあたり，多大なる御支援を頂きました独立行政法人情報通信研究機構 (NICT)，ならびに同大阪リサーチセンター 西浦哲慶氏，山口めぐみ氏，今井佳代子氏に深く感謝いたします．

大阪大学大学院情報科学研究科在学中において御指導，御助言を頂きました大阪大学サイバーメディアセンター応用情報システム研究部門（下條研究室） 秋山豊和講師，野崎一徳氏，坂根栄作特任助教，岡村真吾特任助教，大阪大学大学院工学研究科社会連携室 春本要准教授，兵庫医療大学医学部医学科 加藤精一講師，大阪大学大学院情報科学研究科マルチメディア工学専攻 竹内享助教に深く感謝いたします．

大阪大学大学院情報科学研究科在学中において御支援を頂きました，大阪大学サイバー

メディアセンター応用情報システム研究部門（下條研究室）の秘書の方々，学生，卒業生の諸氏に深く感謝いたします．

参考文献

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Pub, November 1998.

[2] I. Foster and C. Kesselman, *The Grid 2*. Morgan Kaufmann Pub, November 2003.

[3] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Journal of Supercomputer Applications*, vol. 15, no. 3, pp. 200–222, August 2001.

[4] R. P. Hertzberg and A. J. Pope, "High-throughput screening: New technology for the 21st century," *Current Opinion in Chemical Biology*, vol. 4, no. 4, pp. 445–451, August 2000.

[5] M. Orlowski, P. McCoy, F. Gastgeb, K. Appell, L. Ozgur, M. Webb, and J. Burbaum, "Ultra-high throughput screen of two-million-member combinatorial compound collection in a miniaturized, 1536-well assay format," *Journal of Biomolecular Screening*, vol. 5, no. 3, pp. 177–187, June 2000.

[6] N. Yamamoto, O. Tatebe, and S. Sekiguchi, "Parallel and distributed astronomical data analysis on grid datafarm," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004)*, pp. 461–466, November 2004.

[7] JGN II 連携推進部門 テストベッド推進グループ, "JGN II の概要について." `http://www.jgn.nict.go.jp/`.

[8] 史宏宇, 武田伸悟, 長谷川一郎, 伊達進, 水野（松本）由子, 下條真司, "IPv6 におけるセキュアなグリッド環境の構築," 情報処理学会研究報告 2003-HPC-94, pp. 25–30, June 2003.

[9] T. Tashiro, S. Date, S. Takeda, I. Hasegawa, and S. Shimojo, "Architecture of authorization mechanism for medical data sharing on the Grid," in *Proceedings of HealthGrid 2006*, pp. 358–367, June 2006.

[10] T. Tashiro, S. Date, S. Takeda, I. Hasegawa, and S. Shimojo, "Practice and experience of building a medical application with PERMIS-based access control mechanism," in

Proceedings CD-ROM of the 6th IEEE International Conference on Computer and Information Technology (CIT2006), September 2006.

[11] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence, and Interlink Networks, Inc., "Generic AAA architecture," Request for Comments 2903, Network Working Group, Internet Engineering Task Force, August 2000.

[12] 武田伸悟, 伊達進, 下條真司, "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," 情報処理学会論文誌コンピューティングシステム, vol. 45, no. SIG6 (ACS 6), pp. 223–233, May 2004.

[13] 武田伸悟, 伊達進, 下條真司, "グリッドにおける大規模ファイル共有システムの構築," 情報処理学会研究報告 2003-HPC-95, pp. 167–172, August 2003.

[14] 武田伸悟, 伊達進, 下條真司, "グリッドファイルシステム GSI-SFS," 情報処理学会研究報告 2003-OS-93, pp. 97–104, May 2003.

[15] S. Takeda, S. Date, J. Zhang, B. S. Lee, and S. Shimojo, "Security monitoring extension for MOGAS," in *Proceedings of the 3rd International Workshop on Grid Computing & Applications*, pp. 128–137, June 2007.

[16] 武田伸悟, 伊達進, J. Zhang, B. S. Lee, 下條真司, "グリッドにおけるサービス要求フローに着目したセキュリティモニタリング," 第 19 回 コンピュータシステム・シンポジウム (ComSys 2007), vol. 2007, no. 14, pp. 209–216, November 2007.

[17] R. Housley, W. Ford, W. Polk, and D. Solo, "Internet X.509 public key infrastructure certificate and CRL profile," Request for Comments 2459, Internet Engineering Task Force, Network Working Group, January 1999.

[18] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson, "Internet X.509 public key infrastructure (PKI) proxy certificate profile," Request for Comments 3820, Internet Engineering Task Force, Network Working Group, June 2004.

[19] Globus Alliance, "The Globus Alliance." `http://www.globus.org/`.

[20] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *International Journal of Supercomputer Applications*, vol. 11, no. 2, pp. 115–128, Summer 1997.

[21] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," in *Proceedings of IFIP International Conference on Network and Parallel Computing*, pp. 2–13, May 2005.

[22] Open Grid Forum, "Open Grid Forum." `http://www.ogf.org/`.

[23] The EGEE Project, "Enabling Grids for E-sciencE." `http://www.eu-egee.org/`.

[24] The TeraGrid Project, "TeraGrid." `http://www.teragrid.org/`.

[25] BIRN, "BIRN: Biomedical Informatics Research Network." http://www.nbirn.net/.

[26] Earth System Grid, "Earth System Grid (ESG)." http://www.earthsystemgrid.org/.

[27] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch, "A national-scale authentication infrastructure," *IEEE Computer*, vol. 12, no. 33, pp. 60–66, December 2000.

[28] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java commodity Grid kit," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8–9, pp. 643–662, June 2001.

[29] J. Viega, M. Messier, and P. Chandra, *Network Security with OpenSSL*. O'Reilly Media, Inc., June 2002.

[30] A. Bayucan, R. L. Henderson, L. T. Jasinskyj, C. Lesiak, B. Mann, T. Proett, and D. Tweten, *Portable Batch System Administration Guide*, August 1998.

[31] Sun Microsystems Inc., *Sun Grid Engine 5.3 Reference Manual*, April 2002.

[32] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience," *Concurrency - Practice and Experience*, vol. 17, no. 2–4, pp. 323–356, March 2005.

[33] Platform Computing, *Release Notes for Platform LSF(R) Version 6.2*, May 2007.

[34] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A resource management architecture for metacomputing systems," in *Proceedings of IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 62–82, September 1998.

[35] Globus Alliance, "The globus resource specification language RSL v1.0." http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html.

[36] GridSphere Project, "GridSphere Project." http://www.gridsphere.org/.

[37] J. Basney, M. Humphrey, and V. Welch, "The MyProxy online credential repository," *Software: Practice and Experience*, vol. 39, no. 9, pp. 801–816, July 2005.

[38] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A computation management agent for multi-institutional Grids," in *Proceedings of the 10th International Symposium on High Performance Distributed Computing*, pp. 55–63, August 2001.

[39] OpenSCE Project, "OpenSCE project - SCMSWeb." http://www.opensce.org/components/SCMSWeb.

[40] Globus Alliance, "Globus Toolkit Advisories." http://www.globus.org

/toolkit/advisories.html.

[41] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli, *Role-Based Access Control.* Artech House, February 2007.

[42] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke, "Data management and transfer in high performance computational Grid environments," *Parallel Computing Journal*, vol. 28, no. 5, pp. 749–771, May 2002.

[43] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S. Tuecke, "GridFTP protocol specification," GridFTP Working Group Document, Global Grid Forum, September 2002.

[44] M. Horowitz and S. Lunt, "FTP security extensions," Request for Comments 2228, Network Working Group, Internet Engineering Task Force, October 1997.

[45] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney, "File and object replication in data Grids," *Journal of Cluster Computing*, vol. 5, no. 3, pp. 305–314, August 2003.

[46] D. Thain and M. Livny, "Parrot: An application environment for data-intensive computing," *Computing: Practice and Experience*, vol. 3, no. 3, pp. 9–18, September 2005.

[47] J. Bent, V. Venkataramani, N. LeRoy, A. Roy, J. Stanley, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny, "Flexibility, manageability, and performance in a Grid storage appliance," in *Proceedings of the 11th International Symposium on High Performance Distributed Computing*, pp. 3–12, July 2002.

[48] O. Barring, J. Baud, and J. Durand, "CASTOR project status," in *Proceedings of Computing in High Energy Physics*, pp. 365–369, February 2000.

[49] A. Rajasekar, M. Wan, and R. Moore, "MySRB & SRB—components of a data Grid," in *Proceedings of the 11th International Symposium on High Performance Distributed Computing*, pp. 301–310, July 2002.

[50] O. Tatebe, N. Soda, Y. Morita, S. Matsuoka, and S. Sekiguchi, "Gfarm v2: A Grid file system that supports high-performance distributed and parallel data computing," in *Proceedings of the 2004 Computing in High Energy and Nuclear Physics (CHEP04)*, pp. 102–110, September 2004.

[51] J. H. Morris, M. Satyanarayanan, M. H. Conner, J. H. Howard, D. S. Rosenthal, and F. D. Smith, "Andrew: A distributed personal computing environment," *Communications of the ACM*, vol. 29, no. 3, pp. 184–201, March 1986.

[52] M. Satyanarayanan, "The evolution of Coda," in *Proceedings of ACM Transactions on Computer Systems (TOCS)*, pp. 85–124, May 2002.

[53] D. Mazières, *Self-certifying File System.* PhD thesis, Massachusetts Institute of Technology, Boston, May 2000.

[54] K. Fu, M. Kaminsky, and D. Mazières, "Using SFS for a secure network file system," in *USENIX ;login: magazine*, vol. 27, pp. 5–16, December 2002.

[55] M. Kaminsky, G. Savvides, D. Mazières, and M. F. Kaashoek, "Decentralized user authentication in a global file system," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pp. 60–73, October 2003.

[56] M. Carson and D. Santay, "NIST Net: a Linux-based network emulation tool," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 111–126, July 2003.

[57] Hewlett–Packard Company, *Netperf: A Network Performance Benchmark Revision 2.0*, February 1995.

[58] D. Thain, J. Bent, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny, "Pipeline and batch sharing in Grid workloads," in *Proceedings of the 12th International Symposium on High Performance Distributed Computing*, pp. 152–161, June 2003.

[59] Y. Kido, S. Date, S. Takeda, S. Hatano, J. Ma, S. Shimojo, and H. Matsuda, "Architecture of a Grid-enabled research platform with location-transparency for bioinformatics," in *Proceedings of the 15th International Conference on Genome Informatics (Genome Informatics 2004)*, vol. 15, pp. 3–12, December 2004.

[60] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke, "A community authorization service for group collaboration," in *Proceedings of the 3rd IEEE International Workshop on Policies for Distributed Systems and Networks*, pp. 50–59, June 2002.

[61] C. Alfieri, D. Ciaschini, G. Frohner, and S. Lörentey, "VOMS: An authorization system for virtual organizations," in *Proceedings of the 1st European Across Grids Conference*, pp. 33–40, February 2003.

[62] M. Smith, M. Engel, T. Friese, B. Freisleben, G. A. Koenig, and W. Yurcik, "Security issues in on-demand Grid and cluster computing," in *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06)*, pp. 24–32, November 2006.

[63] The Snort Project, *Snort Users Manual 2.8.0*, October 2007.

[64] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23–24, pp. 2435–2463, January 1998.

[65] K. Bauer, *Automating UNIX and Linux Administration.* Apress, September 2003.

[66] H. Ihara and Tripwire Japan, *Tripwire for Linux.* O'Reilly Japan, April 2001.

[67] W. Yurcik, X. Meng, and N. Kiyanclar, "NVisionCC: A visualization framework for high performance cluster security," in *Proceedings of the 2004 ACM Workshop on Visu-*

alization and Data Mining for Computer Security (VizSEC/DMSEC '04), pp. 133–137, October 2004.

[68] R. Byrom, R. Cordenonsib, L. Cornwall, M. Craig, A. Djaoui, A. Duncan, S. Fisher, J. Gordon, S. Hicks, D. Kant, J. Leakec, R. Middleton, M. Thorpe, J. Walk, and A. Wilson, "APEL: An implementation of Grid accounting using R-GMA," in *Proceedings of the UK e-Science All Hands Conference*, pp. 364–366, September 2005.

[69] D. Lim, Q. T. Ho, J. Zhang, B. S. Lee, and Y. S. Ong, "MOGAS: A multi-organisation Grid accounting system," *International Journal of Information Technology*, vol. 11, no. 4, pp. 84–103, August 2005.

[70] B. S. Lee, M. Tang, J. Zhang, O. Y. Soon, C. Zheng, P. Arzberger, and D. Abramson, "Analysis of jobs in a multi-organisation Grid test-bed," in *Proceedings of the 6th IEEE International Symposium on Cluster Computing and Grid Workshops*, pp. 59–67, May 2006.

[71] R. S. Sandhu, E. J. Coynek, H. L. Feinsteink, and C. E. Youmank, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, February 1996.

[72] PRAGMA, "The pacific rim application and grid middleware assembly (PRAGMA)." `http://www.pragma-grid.net/`.