第 10 部

ファイルシステム

第 1 章

はじめに

現在、計算機の高性能化および低価格化が進み、このような計算機群を相互接続するための高速通信技術が一般化しつつある。これに伴い、大学や企業の計算機環境は TSS 環境から開放型分散環境へと移行しており、従来集中して管理されていた資源がネットワーク上に分散して存在するようになった。さらに、広域ネットワークの高速化によって、開放型分散環境の規模はローカルエリアネットワークから広域ネットワークへと拡大されつつある。これに伴い、通信路における大きな遅延、膨大なシステム数といった広域ネットワークの特性を考慮した、開放型分散環境を構成する技術が求められている。

開放型分散環境では、ネットワーク上に分散した資源をローカルノード上の資源と同様に扱うための技術が実用化されている。ファイルシステムはローカルノード上の情報資源をファイルという形態で扱うものであり、分散ファイルシステム (以下 DFS: Distributed File System) はネットワーク上のファイルをローカルノード上のファイルと同様に扱うことを可能にするものである。

UNIX オペレーティングシステムにおいては、データを記憶、共有するための最も基本的な枠組としてファイルシステムが用いられているため、UNIX を用いた広域ネットワーク上での情報共有の最も基本的な枠組として、広域ネットワーク環境に対応した DFS が必要とされている。

これまでに DFS について多くの研究が行なわれ、そのいくつかが実用化されている。しかし、これまでの DFS は広域ネットワークの特性を考慮していない。特に、通信路における輻輳回避機能や、資源に対するアクセスの集中を回避するための負荷分散といった、広域ネットワーク環境での DFS の運用に重要な機能が不足している。また、広域ネットワーク上に分散し、かつ、膨大に存在するファイル資源を DFS で取扱う場合に必要な資源管理機能も不十分である。このような理由から、従来の DFS においてはユーザは広域ネットワークにおける資源アクセスの際に自ら資源管理を行なわねばならない。

これらの問題点を解決するためには、伝送遅延のばらつき、ネットワークの部分的な切断、経路制御の混乱などの障害が起こりやすいという広域ネットワークの特性を考慮した DFS が必要である。

こうした背景をもとに、本研究では Internet のような広域ネットワークにおいて実用に耐えうる DFS として WWFS (World Wide File System) を設計・実装した。本論文では、我々が設計した WWFS における広域での情報資源の管理モデル、WWFS のシステム全体の概要および WWFS クラスタサーバの設計と実装について述べる。

第 2 章

DFS の概念

2.1 UNIX File System の基本概念

現在、多くの DFS の出発点となっているのは Berkeley Fast File System[?] に代表される UNIX のファイルシステム (UNIX File System) である。ここでは DFS を理解する上での前提となる、UNIX File System で用いられている基本概念を簡単に説明する。

- ファイル (file)
  ディスク上に記憶された、初めと終わりを持つ任意長のバイト列のことをファイルと呼ぶ。ファイルは作成時に名前、作成日時などの属性がつけられる。プロセスは (操作の種類, パス名, ファイル名) の対を指定してファイルに対するアクセス権を獲得する。

- パス (path)、ディレクトリ (directory)
  ファイルシステムに情報を蓄積する際、情報を分類するための手段として用いられるのがディレクトリである。UNIX のファイルシステムは情報を木構造に分類することを前提に実装されており (図 2.1)、木構造のルート (root) を起点としてファイルに至る有向辺列上のディレクトリ名をつなげたものをパスと呼ぶ。

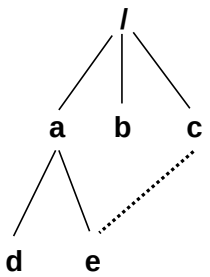

図 2.1: UNIX File System での名前空間。木の節点 (図の a) がディレクトリ、葉 (図の b,d,e) がファイルに対応する。図の c はリンクを表し、e の別名として機能する。

- リンク (link)
  リンクはパス名およびファイル名に別名をつけるための機能である (図 2.1)。

2.2　DFS における基礎的概念

ここでは DFS の議論に必要な用語について説明する。

- 名前空間 (name space)
  ディレクトリ、ファイルおよびリンクにつけられた名前によって構成される空間のことを一般的に名前空間と呼ぶ。ユーザは資源の名前空間上の絶対的あるいは相対的位置を指定することで資源にアクセスする。ディレクトリやリンクが構成するトポロジは DFS によって異なり、木構造をなさない場合もあるため、より一般化された名前空間という言葉を用いる。

- 管理区域 (administrative domain)
  システムの管理方針を決定する際に、合意に達することが比較的容易な組織のまとまりを指す。場合によって異なるが、実際には大学、会社などの一つの機関であることが多い。DFS を広域ネットワークにおいて運用した際に生じる、組織間の管理方針の違い、組織の境界を単位としたアクセス制御の要求などをより一般的に表すために導入された概念である。

- 資源管理 (resource administration)
  DFS での資源管理は、一般に以下のような項目を目的とする。
  - 信頼性の向上。
  - ディスクスペースの有効利用。
  - 通信路の輻輳回避。
  - ファイルサーバにかかる負荷の分散。
  - クライアントに対する応答速度の改善。

- 一貫性制御 (consistency control)
  一般にファイルシステムではパフォーマンスを向上させるため、記憶領域にファイルのキャッシュを持つ。DFS では、クライアント間で共有されたファイルについて一貫性を保つ必要がある。一貫性の基準として何をとるかによって一貫性の意味付けが変わってくるが、代表的なものは以下の通りである。
  - UNIX semantics. すべてのクライアントにおいて、ファイルの最新の状態が見えるという意味で一貫性を持つ。
  - Session semantics. ファイルの open から close までの間、ファイルの内容や状態が他のノードで行なわれた操作の影響を受けないという意味で一貫性を持つ。

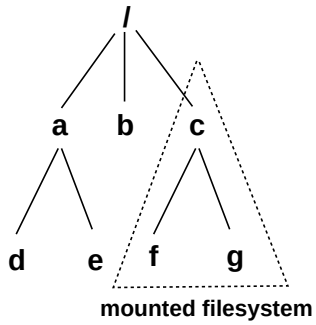

図 2.2: mount は名前空間の合成を行なう操作である。UNIX File System などでは、図のように木のある節点から下に別の木を連接するといったことが行なわれる。

  - Immutable semantics. ファイルが作成された時点からファイルが消えるまで、そのファイルの内容が変わることがないという意味で一貫性を持つ。

2.3 DFS での基本操作

ここでは DFS の基本操作のうち、以後の議論に必要な基本操作について定義する。

- mount
  名前空間のある節点を、別の名前空間の部分集合に連接する操作を mount と定義する (図 2.2)。

- unmount
  mount 操作によって確立された名前空間の連接を取り消す操作を、unmount と定義する。

- lookup
  (directory1, directory2, ..., file) によって構成される資源名から実際の資源への変換を行なう操作を、lookup と定義する。

2.4 DFS の性質に関する定義

DFS の完成度は、以下に定義された性質をどの程度満たしているかによって決まる。

- スケーラビリティ
  DFS のスケーラビリティは、以下の要素の組み合わせとして定義される。

  - サーバを追加することの容易さ。

– クライアントを追加することの容易さ。

- 情報の保護 (Security)
  ある DFS で、クライアント側のノードあるいはユーザによって情報のアクセスを許可あるいは制限することが可能な場合、その DFS は情報の保護ができると定義する。

- 複数の管理区域 (Multiple administrative domains)
  ある DFS を用いて、異なる管理区域との間で相互にアクセスが可能な場合、その DFS は複数の管理区域にまたがるファイルアクセスを実現していると定義する。

- アクセスの透過性 (Access transparency)
  ある DFS を用いて、ネットワークを介した資源へのアクセスをローカルな資源へのアクセスと同様な方法を用いて行なうことが可能な場合、その DFS はアクセスの透過性を持つと定義する。

- 位置の透過性 (Location transparency)
  ある DFS を用いてネットワーク上の資源を要求する際、資源のネットワーク上における位置を知る必要がない場合、その DFS は位置の透過性を持つと定義する。

- 複製の透過性 (Replication transparency)
  ある DFS において、複製された資源に対するアクセスとサーバ上の資源に対するアクセスを区別する必要がない場合、その DFS は複製の透過性を持つと定義する。

- 障害の透過性 (Failure transparency)
  ある DFS において、サーバが故障した時でも代替サーバを用いて作業を続けることができる場合、その DFS は障害の透過性を持つと定義する。

第 3 章

広域分散ファイルシステム

これまで DFS は、LAN などの比較的小規模なネットワークにおいて用いられてきた。しかし従来から、広域ネットワークにおける電子メールなどを用いた情報の共有は非常に活発である。そして今日、開放型分散環境の広域化により、Internet での情報共有の枠組としてより実時間性を備えたものが必要とされている。このような背景から、我々は広域ネットワークにおいても DFS が必要であると考える。

本章では Internet における DFS 利用の現状と問題点について考察し、広域ネットワーク上で DFS を実現する際に解決しなければならない点について考える。

3.1 DFS 利用の現状と問題点

NSFNET[1]から提供されたデータによれば、NSFNET の基幹ネットワークを通るトラヒックのうち約 49%は FTP を用いて転送されたデータである (図 3.1)。また University of Colorado のゲートウェイで測定されたデータでは、anonymous FTP を用いたファイル転送トラヒックのうち 54%は重複したファイル転送である [?]。

これらのデータから次のことが言える。

- Internet でファイルを共有する際、現在もっとも広く用いられているのは FTP である。
- FTP は異なるユーザ間でのファイル転送に関してキャッシュを行なわないので、通信路が十分に生かされていない。
- 広域ネットワークでのファイル共有の枠組として、DFS は用いられていない。

これらの状況は、前章で述べた DFS の問題点によるものであると考えられる。それらをまとめると次のようになる。

- FTP はアクセスの透過性をもたない。
- FTP, NFS 以外の DFS は、既存のソフトウェアとの親和性が良くない。

[1]米国 National Science Foundation によって設立・運営されている学術研究用ネットワーク

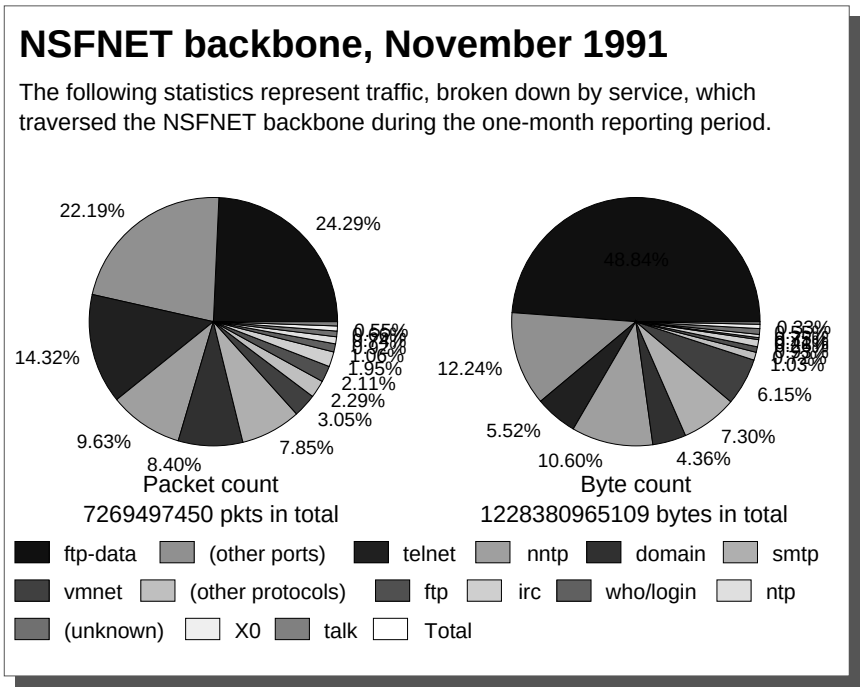

図 3.1: 1991 年 11 月からの一か月間、NSFNET 基幹ネットワークにおいて測定されたトラヒックデータをサービス別に分類したもの。

- NFS, AFS, PFS のいずれにおいても効果的なキャッシュの機構が導入されていない。

また従来の DFS には、以下の特徴が共通にみられる。

- 伝送遅延にばらつきがある場合を考慮していない。
- 伝送遅延が 500msec を越えるような通信路での使用を想定していない。
- Internet は常に部分的に切断されているということが設計段階から考慮されていない。
- 通信路に障害が起こった場合を設計段階から考慮に入れていない。

以上のような考察から、既存の DFS は広域ネットワークへの適用が困難であると考えられる。

3.2 広域での DFS 実現の障害

DFS は以下の特性を持つことが望ましいとされている。

- スケーラビリティ
- 情報の保護
- 複数の管理区域
- アクセスの透過性
- 位置の透過性
- 複製の透過性
- 障害の透過性

広域ネットワークにおいてこれらの要求をすべて満たし、かつ一貫性制御を厳密に行なうことは非常に困難である。またこれらの要求を不完全に満たした DFS を広域で運用した場合、LAN 環境では問題とならなかったような欠陥が致命的なものとなり、ファイルの損失やネットワークの輻輳をもたらしたり、クライアントを操作不能な状態に陥れたりする。

しかしこれらの要求は、本来 LAN でのファイルシステムに対するものであり、これらが広域分散ファイルシステム[2]においても必要かどうかは検討の余地がある。したがって以下の節では、情報共有のモデルを考え直すことで広域分散ファイルシステムに必要な機能について検討する。その後に、それらの機能を備えた広域分散ファイルシステムの実現可能性について考察する。

[2]広域ネットワークにおいて実用に耐えうる DFS を指す。

3.3　情報共有のモデル

情報の共有を行なう状況としてはさまざまな場合が考えられるが、ここでは最も典型的な二つの状況について考察する。

3.3.1　「密な」情報の共有 (tight sharing of information)

「密な」情報の共有とは、情報を頻繁に更新しながら共有している状況を指す。身近な例としては、会議やミーティングが挙げられる。計算機を用いた密な情報共有の例としては、ソフトウェアの共同開発が挙げられる。

ファイルシステム上で密な情報の共有を行なうためには、ファイルの最新の状態をネットワーク上のノードに反映させるための仕組みが必要である。ファイルをキャッシュしている場合には、厳密な一貫性制御が必要とされる。なお、広域ネットワーク[3]ではこういった仕組みは実現不可能であることが理論的に証明されている [?]。

3.3.2　「疎な」情報の共有 (loose sharing of information)

「疎な」情報の共有とは、情報の更新がまれにしか行なわれないときに情報を共有している状況を指す。身近な例としては、書籍の出版が挙げられる。計算機を用いた疎な情報共有の例としては、オンラインでテクニカルレポートなどの文書を配布する場合が挙げられる。

このような状況では、密な情報共有の場合のような厳密な実時間性は要求されない。したがって、ファイルシステムの上で疎な情報の共有を行なう場合には厳密な一貫性制御は必要とされない。

3.4　広域分散ファイルシステムに求められる機能と実現可能性

Internet におけるファイル共有の形態を見る限りにおいては、広域分散ファイルシステムにおいて密な情報共有が行なわれることはないと考えられる。また、あるファイルに対する読み出しが複数のノードから行なわれる場合、そのファイルが更新される確率は非常に小さいことが統計的に知られている [?]。疎な情報の共有を行なう場合だけに限れば、一貫性制御を行う必要がないため先に述べた要求を満たすような広域分散ファイルシステムの実現可能性は極めて高いと我々は考える。

また、Internet における DFS 利用の現状に関する考察から、以下のような特性が広域分散ファイルシステムに求められていると考えられる。

- 既存のソフトウェアとの親和性が良い。

[3] 正確には、有界だが不確定なメッセージ伝達時間を持つシステム

- 既存のファイルシステムと同じ方法でアクセスできる。
- 広域でのキャッシュの機構を持つ。
- 管理区域内部で発生するファイルアクセスを統制できる。

また我々は以下の三つの属性を、広域ネットワークで実用に耐えうる DFS を実現するために不可欠なものであると考える。

- 伝送遅延にばらつきがある場合でも正しく機能する。
- 伝送遅延が 500msec を越える通信路でも正しく機能する。
- 常に部分的に切断された状態にあるネットワークにおいても、正しく機能する。

本研究では以上のような考えに基づいて、新しいファイルシステムの設計・実装をはじめている。

第 4 章

広域分散ファイルシステム WWFS

本研究では、「疎な情報の共有」を行なうための枠組として、WWFS (World Wide File System) を設計・開発している。WWFS ではネットワーク上のファイルに対する操作を提供するだけでなく、広域ネットワークにおける資源の発見、資源へのアクセスを含めたシステムを実現する。そのために我々は、広域分散ファイルシステムにおける資源管理の新しいパラダイムを構築し、これに基づいて実装をすすめている。

本章では WWFS の構成について、いくつかの視点から概観する。

4.1 WWFS のプロセス構成

一般に DFS は、一つまたは複数のプロセスから構成される。もっとも単純な DFS はファイルサーバとクライアントだけから構成される。しかしこの形態を広域ネットワークに拡張した場合、トラヒックの集中、アクセス時間の増大が予想される。WWFS ではシステムの成長にかかるコストをできるだけ小さく抑えるため、資源アクセス、アクセスの最適化、記憶管理、資源管理の機能を分離し、以下のような構成とした (図 4.1)。

- クライアント (Client)
  ファイルを使用する任意のプロセス。クライアントがネットワーク上のファイルに対してアクセスする際、クラスタサーバに対する要求に変換される。

- クラスタサーバ (Cluster server daemon)
  頻繁にアクセスされるファイルをキャッシュすることで、アクセス時間を改善するサーバプロセス。クライアントの要求するファイルあるいはファイル属性がキャッシュになければ、要求をファイルサーバに取り次ぐ。

- ファイルサーバ (File server daemon)
  ファイルという資源を保存し、クライアントからの要求に応じてそれを配布するためのプロセス。

- 資源データベース (Resource database daemon)
  クライアントの要求する資源がネットワーク上のどのサーバにあるかを記録し、クライアントに知らせるためのプロセス。

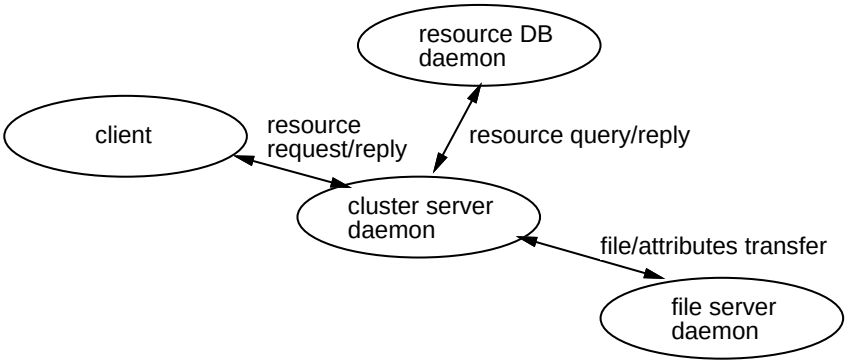

図 4.1: WWFS での階層的なプロセス配置。

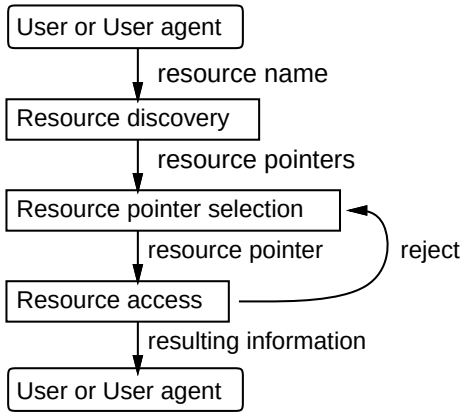

図 4.2: 広域ネットワーク環境における資源利用のフェーズ。

4.2 資源アクセスのフェーズ

広域ネットワーク環境における資源の利用は、一般に以下のようなフェーズから構成される (図 4.2)。

1. Resource discovery
   利用者が目的とする資源の発見を行なう。広域ネットワークにおいては、膨大な資源が分散して存在する。そのため、資源発見のための効率のよい手段を提供する必要がある。WWFS では、資源のデータベースに対する問い合わせを行なうことによって資源へのアクセス手順 (resource pointer) に関する情報を得る。

2. Resource pointer selection
   Resource discovery で得られた情報では、複数の資源が候補として得られる可能性

がある。この場合、利用者はアクセス時間、バージョンなどの条件により最も適切と思われる資源を選ぶ。ここでは、アクセス権限に関しては評価を行なわない。

3. Resource access
最終的に得られた資源に対して、アクセス手順にしたがってアクセスを行なう。アクセス権限の評価によってアクセスが拒絶された場合には、resource pointer selection のフェーズに戻り、新たな候補を選ぶ。

WWFS では上記のフェーズをそれぞれ独立したプロセスとして実装し、プロセス同士の連係はネットワークを介した手続き呼び出し (Remote Procedure Call) を用いて行なう。

4.3 Filesystem interface

WWFS 上のファイルは、従来のファイルシステムにおける書き込み操作が一切できないことを除けば、従来の UNIX のファイルと同様に扱える。WWFS の filesystem interface は以下の特徴を持つ。

- 読み出し操作のみである。いかなるセキュリティ違反があっても、ファイルが改変されることはない。
- アクセス制御リスト (ACL: Access Control List) を用いてアクセスを許可あるいは制限することが可能である。
- ACL の内容が持つ意味は、ACL を評価する認証システムによって決まる。
- ディレクトリと ACL は対になっている。つまりアクセス制御はディレクトリ単位で行なう。
- ディレクトリと authentication ticket は対になっている。つまり認証はディレクトリへの最初のアクセスの際に行なわれる。Authentication ticket の有効期間は認証モジュールによって決定される。

4.4 Submission interface

ファイルを共有するため、ファイルサーバに一意な (名前, 属性) の対を持つファイルを新たに置くことを submission と定義する。Submission interface はネットワークを介して submission を行なうためのインターフェースである。これにより、書き込みが可能なファイルシステムで起こるキャッシュの整合性の問題、操作の直列化の問題は、WWFS では扱う必要がない。WWFS の submission interface は以下の特徴を持つ。

- ファイルの更新操作しか提供しない。Submit されたファイルを変更することはできない。

- Resource database との円滑な連係処理。資源の Submission の段階において提供された情報を resource database へ登録し、資源に関する情報として有効利用をはかる。

- ユーザがファイルの属性を自由に定義できる。各々の属性の名称や意味はそれを使用するプログラムによって決まる。いくつかの属性の使い方、形式、意味を予め定義することで、それらを資源管理の局面において有効に利用することが可能であると考えられる。

4.5 WWFS での資源管理

WWFS では意味的にまとまりをもった複数のファイルを「ボリューム」として捉え、ボリュームを単位とした資源の管理を行なう。例えば資源の複製はボリュームを単位として行なわれる。Resource database ではユーザ定義属性 (UDA: User Defined Attributes) を有効に利用することで、資源の同一性の判定を効率的に行なうことが可能である。例えば、以下のような UDA が考えられる。

- path
  複製された資源には属性「path」を定義し、資源が複製される過程を記録する。

- version tree
  同一ソフトウェアの異なるバージョンが存在する場合には属性「version tree」を定義し、ファイルの変更履歴を記録する。

- cache policy, expire policy
  キャッシュのポリシ、ファイル消滅 (expire) のポリシ等について特別な扱いが必要な場合は、「cache policy」、「expire policy」などの属性を定義し、キャッシュを行なうプロセス、ファイルの消滅を行なうプロセスに対する指示とする。

4.6 WWFS での名前空間

WWFS が提供する名前空間は、以下の二つの名前空間からなる (図 4.3)。

- 各ボリュームが提供する名前空間。ファイルサーバによって提供される名前空間である。

- 全体の名前空間。各ボリュームによって提供される名前空間を動的に連接することで、全体の名前空間を構成する。

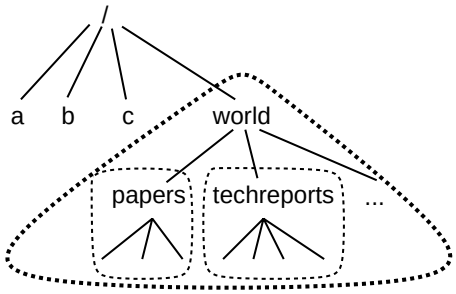

図 4.3: WWFS における名前空間。「papers」「techreports」が提供する名前空間をディレクトリ「world」の下で動的に連接し、全体の名前空間を構成している。

WWFS では現在使われているボリュームだけを mount することで、不必要なボリュームの存在をユーザから隠蔽し、ユーザの混乱を防ぐ。またボリュームを自動的に mount, unmount する機構を用いて、障害に対する透過性を実現できる。

4.7　本研究の位置付け

本研究では WWFS における資源利用の最終段階である resource access のフェーズにおいて必要な技術を設計・実装している。

Resource access を行なうためにはクライアント、クラスタサーバ、ファイルサーバの三つのプロセスが実装されていなければならない。我々はクライアントのプロトコルとして Sun NFS、ファイルサーバのプロトコルとして FTP を採用した。こうすることで、既存のソフトウェアとの後方互換性を保つことが可能となり、開放型分散環境において相互操作性を獲得できると考えられる。ファイルサーバ、クライアントを構成するソフトウェアは既存のものが利用できるので、本研究では両者の間に位置するクラスタサーバを実装する (図 4.4)。

クラスタサーバを実装することによって以下の目標を達成することができる。

- 広域での重複するファイル転送要求をクラスタ単位で最適化する。
- 管理区域内部で発生するファイルアクセスを統制できる。
- クライアントからの広域ネットワークに対する透過的なアクセスを可能にする。
- 資源から位置情報を分離することによって資源の複製が容易となり、広域ネットワークの部分的な切断に対して柔軟に対処できる。

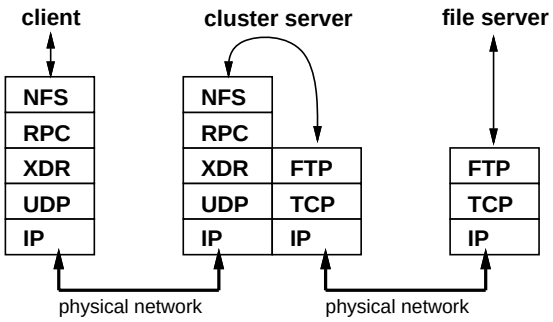

図 4.4: プロトコルスタックでみたクライアント、クラスタサーバ、ファイルサーバ間の関係。

第 5 章

WWFS クラスタサーバの設計と実装

本章では、WWFS クラスタサーバの設計と実装について述べる。

5.1 クラスタサーバの設計

5.1.1 スケーラビリティの実現

WWFS クラスタサーバでは、高いスケーラビリティを実現するために以下の技術的目標を設けた。

1. クライアント単位でのサーバへの負荷の増加を最小限にとどめる。
2. 単位時間当たりに処理するリクエストの数を最大にする。

これらの目標を実現するため、

- 同一サーバへの接続はできる限り共有する (目標 1)
- ファイル及び属性のキャッシュの有効時間を非常に長くする (目標 1)
- 可能な限りプロセススイッチを防ぐ (目標 1)
- ファイルサーバおよびクライアントとの同期を防ぐ。つまり、TCP, RPC などのネットワークに対する入出力はブロックせずに行なう (目標 2)

といった実装技術を用いた。

5.1.2 複数の管理区域にまたがるアクセスの透過性の実現

WWFS のクラスタサーバに対しては、NFS のファイルサーバとまったく同様にアクセスできる。したがって WWFS は高いアクセス透過性を実現している。また実際のファイル転送には、複数の管理区域に対応したファイル転送プロトコルを用いることで、複数の管理区域にまたがるファイルアクセスを容易に行なうことができる (図 4.4)。

WWFS クラスタサーバでは、ファイル転送プロトコルの動的な選択が可能な構成をとっている。現在、使用可能なファイル転送プロトコルは FTP のみである。

5.1.3 その他の透過性の実現

WWFS は、位置の透過性、複製の透過性、障害の透過性を高い水準で実現するために以下のような方針をとった。

- ボリューム名を導入することで、資源名を位置情報から分離した。
- 動的にファイルサーバを選択することで、複製された資源を有効利用する。
- 定期的にファイルサーバの状態を監視することで、障害発生時にサーバを切り替えることが可能である。

5.2 クラスタサーバの実装

5.2.1 構成

クラスタサーバは以下のモジュールからなる。

- インターフェース部
  - NFS とのインターフェース
  - 資源データベースとのインターフェース
  - ポリシに基づいてサーバ、プロトコルを選択するモジュール
- ファイルシステム部
  - キャッシュファイルシステム
  - エラーファイルシステム
  - ルートファイルシステム
  - プロトコル別ファイルシステム
- マイクロカーネル部
  - イベントディスパッチャ
  - 時系列を単位とするスケジューラ
  - タスクを単位とするスケジューラ
  - ロック管理モジュール
- 内部情報管理部
  - 接続管理モジュール

- ディレクトリ管理モジュール
- ファイル管理モジュール
- サーバ管理モジュール
- ボリューム管理モジュール

このほかに今回作成したプログラムは以下の通りである。

- mount を行なうプログラム
- unmount を行なうプログラム

5.2.2　インターフェース部

インターフェース部では WWFS を構成する他のプロセスやクライアントとの通信を行なう。

NFS クライアントとの通信部分は Sun Microsystems が開発した rpcgen[?] を用いて自動生成した。資源データベースがまだ実装されていないので、資源データベースとのインターフェースはファイルを読み込むだけの簡単なものを実装した。ポリシに基づくサーバ選択、プロトコル選択は、ネットワークの構成やファイルサーバの負荷に基づいたコストの詳細な評価が必要なので、インターフェースの定義だけにとどめている。

5.2.3　ファイルシステム部

ファイルシステム部では、ファイル転送プロトコルとファイルシステムのインターフェースの実装にかかるコストを最小化するため、以下のような構成をとっている (図 5.1)。

- キャッシュファイルシステム
  キャッシュにヒットした場合、実際のファイル転送に関連する処理は行なわず、クライアントの要求を処理する。

- プロトコル別ファイルシステム
  キャッシュにヒットしなかった場合、実際のファイルやファイル属性の転送を行なう。現在 FTP プロトコルを用いたものが実装されている。今後、NFS over TCP を用いたものの実装を予定している。

- ルートファイルシステム
  クライアントから見た名前空間を構成するためのファイルシステム。ルートファイルシステムの名前空間は動的であり、以下の規則に基づいて構成される。

  - あるボリュームがルートファイルシステムの名前空間に存在しなければ、その時点でルートファイルシステムに mount される。

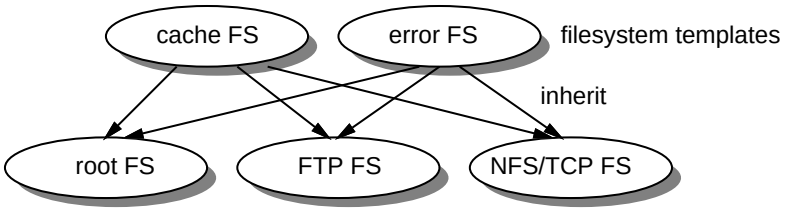

図 5.1: WWFS での階層的なファイルシステム構成。ファイルシステムに共通の機能を cache FS, error FS にまとめ、実際の名前空間を構成する部分とプロトコルに依存する部分を各々のファイルシステムで記述している。

  - あるボリュームがルートファイルシステムに mount されていて、一定期間アクセスがなければ、ルートファイルシステムから unmount される。

- エラーファイルシステム
  クライアントにエラーを返すファイルシステム。クライアントの要求が不適当なものであった場合、エラーファイルシステムに処理が移される。

5.2.4 マイクロカーネル部

UNIX オペレーティングシステムはプロセスを単位としたスケジューリング、ロック、ディスパッチの機構を提供しているが、WWFS クラスタサーバでは以下の理由からこれらを一つのプロセス内部で実現する必要があった。

- スケーラビリティ。クライアントが増加に対してサーバプロセスの増加を抑えるためには、一つのプロセスで複数の TCP 接続を処理する必要がある。このため、プロセス内部でのスケジューラが必要である。

- Sun RPC との整合性。Sun RPC でのリクエスト再送機構は LAN 環境を前提としたものである。広域ネットワークのような伝送遅延にばらつきがある環境において Sun RPC を機械的にファイル転送プロトコルに変換すると、ネットワークに対して大きな負荷をかけることになる (図 5.2)。したがってファイルあるいはディレクトリなどの資源を単位としたロックの機構が必要になる (図 5.3)。

- 複数のプロセスで、接続の共有、ロックの共有を行なうことは難しい。これを実現することは不可能ではないが、頻繁なプロセススイッチはパフォーマンスの低下をもたらし、スケーラビリティ実現の大きな妨げとなる。

- イベントディスパッチャ
  複数のクライアント及びサーバとの通信はすべて、イベントディスパッチャで受け取

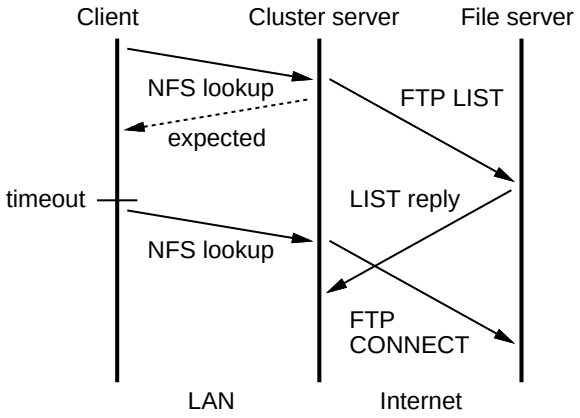

図 5.2: Sun NFS から FTP に機械的にプロトコルを変換した場合のタイムラインを表す図。FTP プロトコルでは、一つの接続を用いて同時に複数のリクエスト・リプライを処理することができないので、リクエストの再送時に新たな接続を行なってしまう。破線の矢印は NFS クライアントが期待するリプライを表す。

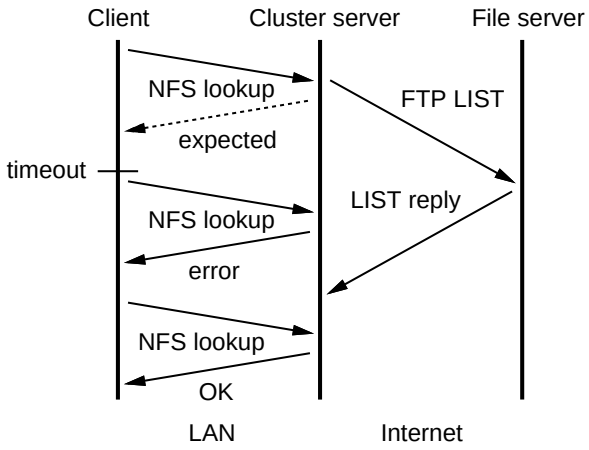

図 5.3: ロックを用いた場合のタイムラインを表す図。ロックを用いて、再送されたリクエストに対してはエラーを返すようにする。

られ、対応するモジュールへの関数呼び出しに変換される。これは特定のクライアントあるいはサーバとの同期を防ぐためである。タイムアウトその他のネットワークを介さないイベントの通知もすべてここで受け取られ、処理される。

- 時系列を単位とするスケジューラ
  クライアントに対する返答、メモリ上の情報のガベージコレクション、通信路の状態の監視など、時間を基本とした処理をスケジュールするために用いられる。これは UNIX オペレーティングシステムにおける callout に相当する。

- タスクを単位とするスケジューラ
  本来ある状態にとどまって操作の完了を待つような状況で、操作の完了を待たずに処理を続行する場合に用いられる。具体的には、(待ち事象, 呼び出される関数, 変数の束縛) の対を登録し、操作の完了時に指定された関数を呼び出す。これは UNIX オペレーティングシステムにおける wait channel に相当する。

- ロック管理モジュール
  メモリ上の任意の情報について、ロックを設定あるいは解除する機構を提供する。

5.2.5 内部情報管理部

内部情報管理部は、スケーラビリティを実現するために、クラスタサーバの高速化、大規模な資源への対応を重視して実装されている。

- 接続管理モジュール
  サーバに対する接続の状態を管理する。接続を集中的に管理することによって、次のような利点が得られる。

  – 同一サーバに対するファイルアクセスに必要な接続の本数を削減することができ、ファイルサーバの負荷が軽減される。
  – 新たに接続を確立しなければならない場合が減少し、クライアントに対する応答速度が改善される。

  サーバの負荷を軽くするため、一定の時間使われなかった接続は閉じられる。

- ディレクトリ、ファイル、ボリューム管理モジュール
  これらは各々の情報のキャッシュを管理する。キャッシュはディスク上に保存し、その一部をメモリ上に持つ。メモリ上の情報が肥大してパフォーマンスが低下することを防ぐため、一定の時間使われなかった情報はメモリ上から解放される。長期間使われなかった情報はディスク上から消去される。

- サーバ管理モジュール
  サーバに関する情報を管理する。現在はサーバ名、サーバとの接続、サーバとの接

続に用いるプロトコルといった情報しか管理していないが、近い将来、サーバとのファイル転送やサーバ状態の監視を通じて得られた統計的情報を蓄積し、サーバ選択部に対するフィードバックを行なえるよう拡張する予定である。

第 6 章

WWFS クラスタサーバの評価

6.1 実装状況

クラスタサーバの基本的な部分の実装は完了し、WIDE Internet の基幹ネットワークを用いて試験的に運用を行なっている (図 6.1)。ただし、資源データベース、認証モジュールなど、まだ実装されていない部分に対しては、インターフェースの定義にとどめている。

6.2 クラスタサーバの実装で得たもの

クラスタサーバの実装を行ない、NFS サーバの挙動を解析していく過程で、以下のことが明らかにされた。

- 広域ネットワークの基幹リンクでは伝送遅延に大きなばらつきがある。
- 遠隔手続き呼び出しの機構では、呼び出し側がサーバからの返答を必ず待つため、並列性をある程度もつクライアントでもブロックしてしまう。

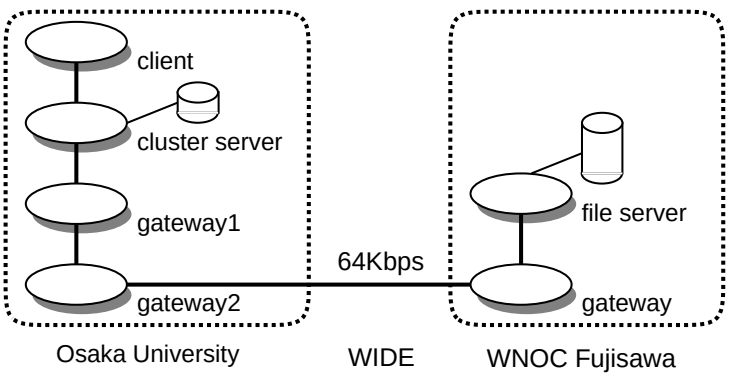

図 6.1: WIDE 基幹ネットワーク上で WWFS クラスタサーバを試験的に運用している。

6.3　目標の達成度について

アクセスの透過性、位置の透過性はかなり高い水準で達成できた。スケーラビリティ実現のための技術はすべて実装したので、高いスケーラビリティが実現されている。これについてはクラスタサーバを実際に運用してデータを収集する必要がある。複数の管理区域にまたがる運用が可能であることは、WIDE Internet の基幹ネットワーク上での運用を通して実証した。

動的なファイルサーバの選択については、ポリシに基づく意志決定部分が実装されていないため検証されていない。障害発生時のサーバの切り替えについては、現在実装を行なっている。

第 7 章

おわりに

本論文では広域ネットワークにおいてファイルシステムに求められる機能について考察し、それらの要求を満たす新たな資源共有の枠組「WWFS」の概要、システム構成、およびその基礎となる資源アクセスの機構の設計と実装について述べた。

本研究では、ネットワーク環境における情報共有モデルとして、密な情報共有のモデルと疎な情報共有のモデルを提案し、広域ネットワークにおける DFS では疎な情報共有のモデルに基づいた構成が効果的であることを示した。さらに、それに基づいた WWFS の設計と実装を行ない、試験的運用を開始している。

しかしながら、WWFS のクラスタサーバ以外の部分はまだ実装されておらず、既存のファイルシステムをはじめとするネットワークサービスに依存しており、今後はそれらに代わるより優れたネットワークサービスを設計・開発していく予定である。また WWFS には、以下のような研究・開発が今後必要であると考える。

- **広域でのアクセス制御**
  従来の広域でのファイル共有では、Anonymous FTP に見られるようにアクセス制御が行なわれないのが一般的である。しかし広域ネットワークでのファイル共有においても、アクセスのきめ細かな制限あるいは許可を行なうことが必要である。今後、広域ネットワークにおけるアクセス制御に対する要求分析、アクセス制御モデルの詳細化を行なう必要がある。

- **認証システムとのインターフェースの実装**
  認証システムは、アクセス制御を行なうための基礎となる部分である。WWFS では、認証システムとして SPLICE/AS[?] を用いる予定である。

- **トラヒックの測定**
  WWFS の成否を客観的にとらえるため、WWFS を導入することによる広域ネットワークのサービス構造の変化を観測する必要がある。このため、基幹ネットワークと大阪大学学内ネットワークを結ぶゲートウェイの近傍において、トラヒックを長期にわたって継続的に監視する。

- **広域における資源管理の方法の確立**
  WWFS は広域での柔軟な資源管理の枠組を提供するもので、その中での資源管理

のあり方について示唆するものではない。今後、我々は学内ネットワークにおいて WWFS を運用し、広域ネットワークにおける新たな資源管理の方法を検討する必要がある。

- **Internet における運用に向けて**
  広域ネットワークにおける資源管理の方法を確立し、学内ネットワークにおいて検証を行なった後、これを WIDE Internet などの実際の広域ネットワークにおいて運用し、さらに検証を行なう。