移動計算機環境のための
分散システムに関する研究

宅間 啓
慶應義塾大学大学院政策・メディア研究科

指導教員:
村井 純
徳田 英幸
楠本 博之

平成12年1月

修士論文要旨 1999 年度 (平成 11 年度)

移動計算機環境のための分散システムに関する研究

世界中の計算機がインターネットによって資源を共有し始めている。広域分散ネットワークであるインターネット上の資源を適切に共有する計算機環境を実現するためには分散システムが必要である。また、移動型計算機が普及しているため、ネットワーク接続性が保証されない計算機を考慮した情報共有を実現しなくてはならない。

本研究では、移動型計算機利用環境のための分散システムについて考察し、システムの設計を行なった。そして具体的なシステムとして、耐故障性をもった分散型ファイルシステムである FTFS(Fault Tolerant File System) を実現した。FTFS では、分散環境の中で情報共有時にネットワーク接続性が確保できない場合を考慮し、リビジョン管理を徹底することにより整合性を保持する。

例えば、FTFS を使ってファイルを共有する場合、ファイルを共有する際に新しいリビジョンを生成し、その状態を管理する。同じファイルを別の場所で共有する際には、編集中の状態でない最新のリビジョンから新しいリビジョンを生成し、その状態を管理する。リビジョンの管理を徹底することにより、ネットワークの接続性が変化した場合でも、その変更履歴が記録され情報の整合性が保証される。

また、FTFS は既存のシステムとの親和性が高いため、多くのプラットフォームでの動作が可能である。分散計算機利用環境を実現するためのアプローチとして、オペレーティングシステムを作るといった手法もある。しかし、既に世界中の計算機上で動いているすべてのオペレーティングシステムを変更することは困難であり、実現することは難しい。FTFS は既存の多くのプラットフォームでの動作を考慮し、サーバプログラムとライブラリにより実現した。

本研究では、ネットワークの接続状態が変化する計算機が利用することにできる分散計算機利用環境として、リビジョン管理機構を持つファイルシステム FTFS を実現した。本システムにより、分散環境下での情報共有において変更履歴が保存され、状態が管理される。この機能により、ネットワーク接続性が保証されない移動型計算機の利用を含む、分散計算機利用環境を実現した。

キーワード

1. 分散環境、2. 耐故障性、3. ファイルシステム、4. 移動型計算機、5. バージョン管理

慶応義塾大学 大学院 政策・メディア研究科

宅間 啓

Abstract of Master's Thesis

Academic Year 1999

Study on Distributed Computing Environment with Mobile Computers

Various computer system resources in the world is being shared. Resources on wide area distributed network, including the Internet, is well integrated by distribted systems. Mobile computer must be thought of.

In this research, a distributed system for the mobile computing environment is discussed and desiged. A distributed fault tolerant file system, FTFS(Fault Tolerant File System), is implemented. There are many cases that mobile host cannot keep its connectivity to the network. In those cases, FTFS achieves consistency of the information by a tight revision management mechanism.

When a file is shared by FTFS, a new revision of it is created and the state of it is managed by FTFS. If the same file is shared with multiple places, a new revision of the file is created from the unedited revision of the corresponding file. In this research, a moblie computer with unstable network connectivity is enabled. FTFS enables revision management, within it's file system. This function enables use of file system with unstable network.

Keywords :
1.distributed system, 2.fault tolerant, 3.file system, 4.mobile computer, 5. version control

Keio University Graduate School of Media and Governance

TAKUMA Kei

目 次

図 目 次

第1章　序論

1.1　背景

1.1.1　分散計算機利用環境

計算機利用は生活の一部になりつつある。計算機はレポートを作成するために使い、統計情報を計算するために使う。学生は教官との連絡に電子メールを使い、疑問は電子ニュースやWWWなどを使って解決することができる。通信販売、就職活動、手紙の代わり、新聞の代わり、ステレオの代わり、娯楽のためなど、生活の中で行っていることの多くのことを、計算機を使ってできるようになってきている。

我々の生活は、ネットワーク時代に入っている。例えば、大学はキャンパスネットワークを整備している。学習や研究を支援する目的にとってもネットワーク は有益である。

キャンパス内で計算機を使える場所は一箇所ではない。さまざまな教室や図書館等に配置されている。学生は特定の計算機を利用するのではなく、分散して配置してある計算機のどれかを任意に選んで利用することができる。キャンパス内のどの計算機を使っても同じ計算機利用環境が使えるように整備されている。このような環境を実現するための技術が分散システムである。

初期にはネットワークは計算機と計算機の間での情報のコピーを行うために使われた。やがて、遠隔地に設置されている計算機をネットワークを通して使うようになった。そして、ネットワークで相互に接続されている計算機を、ソフトウェアによってまるで一台の計算機であるかのように、透過的に扱えるようになった。このような分散透過性の実現が分散システムの大きな特徴である。

分散システムによってユーザは異なる計算機から常に同じ計算機利用環境を使うことができる。計算機利用環境はユーザにとって自分の机に例えられる。そこには、過去に読んだ参考資料が保存されていたり、届いた手紙を整理してある箱があったり、書いている途中の原稿用紙がしまってある。自分の机に何を置き、それをどこに置いておくのかを自分で決めることができる。計算機利用環境も同じように個人ごとに用意される計算機利用環境であり、電子情報を好きな所に保存しておくことができる。

自分の机がいくつもあった場合を考えると、計算機利用環境が複数ある場合の不便さが容易に想像できる。今必要なノートはどの机にしまってあるのかを覚え

ておかなければならない。さらに今座っている机にないものはわざわざ席を立って取りに行かなければならない。計算機利用環境が複数ある場合も同様である。異なる計算機利用環境 にしかない情報は、それらの計算機利用環境の間で情報を移動しなければならない。

分散システムを使うとこのような不便さを解決することができる。計算機利用環境は計算機上に作られた仮想的な世界である。分散システムを使って、物理的に異なる計算機上にある計算機利用環境を一つの計算機利用環境であるかのように見せることができる。

1.1.2 移動型計算機

キャンパスネットワークを用意するのと並行して、大学は学生に移動型計算機を使うことを奨励している。キャンパスネットワーク上には学生よりもはるかに少ない数の計算機しか用意できないからである。学生の生活が計算機と密接な関係を持つようになるにつれて、計算機の数は不足する。大学が用意する計算機を増やすことは、設置する場所など物理的な制約がある。このような問題を解決するためにも一人一人が移動型計算機を持つことは適切な解決策である。移動型計算機を持っていれば、ユーザがどこにいようと計算機を利用できる。生活が計算機と密接なかかわりを持つにつれて、計算機を持ち歩く必要性は高くなっていく。

計算機が生活に深くかかわるようになった一つの理由は、計算機が様々なメディアを統合するメディアであるからである。計算機は数値計算などのプログラムを実行するだけの純粋な計算機システムから発展し、音や映像や文字を自由に組み合わせることができ、インタラクティブ性のある情報を提供できる。

これらの機能を提供するためには、計算機が高い性能を持っていなければならなかった。技術の進歩によって、計算機が実用上十分な性能のまま小型化された。液晶画面も密度が高く、小さくても高い解像度のものを使えるようになった。小型ディスク装置の容量も多くなっている。

こうして、移動型計算機が広く使われるようになった。移動型計算機は、ノート型計算機やラップトップ計算機だけでなく、機能を限定した PDA 等も含む概念である。移動型計算機とはユーザが持ち歩くことを前提にした、様々な種類の計算機のことである。

移動型計算機は安価で高性能になっているので、移動型計算機内部で閉じた作業環境を作ることができる。しかし、電子メールや電子ニュース、WWW の閲覧など、ネットワークを必要とする作業を行うためにはネットワークに接続する必要がある。この目的のために、例えばキャンパスネットワークでは移動型計算機をネットワークに接続するための設備を用意している。

計算機が社会に浸透したもう一つの理由は、コミュニケーションを支援する機能を持つからである。この機能は計算機だけでなく、通信手段としてのネットワークの普及が大きな要因である。

ネットワークによって計算機は相互に接続されている。ネットワークを扱うために計算機上のシステムは様々な改良が加えられて来ている。今日のインターネットを支えているのは OSI 参照モデルで定義されているそれぞれの層での技術革新である。

移動型計算機でも、届いた電子メールを印刷したり、プログラムで処理したいという要求はあるだろう。機能が限定されている移動型計算機であっても、自分の計算機利用環境の一部として使われるべきであり、物理的にわかれているところを透過的に扱える必要がある。

現在移動型計算機を分散計算機利用環境の一部として利用する環境は実現されていない。移動型計算機はその性質上、常にネットワークへの接続性を維持できるわけではない。また、移動した先の管理ドメインが異なる場合もある。移動型計算機を位置固定型計算機のように使うことで分散計算機利用環境の一部にすることはできる。しかし、それでは移動型計算機の長所を捨ててしまうことになる。

一般的に移動型計算機において、要求される機能は位置固定型計算機に比べて多い。位置の移動にともなって計算機利用環境も大きく変化するからである。移動型計算機に求められる機能には、以下があげられる。

- 通信環境の動的な変化に対応する機能
  - ネットワークの切断の検知
  - ネットワークの接続の検知
  - 利用可能なネットワークの変化の検知
- 消費電力の動的な変化に対応する機能
  - 電力供給源の変化の検知
  - 電力の残量の監視
- システムの突発的な停止に対応する機能

移動型計算機はネットワークに接続できる。ネットワークに接続する場所も移動することができる。移動型計算機を接続した場所に応じて、その環境に用意されている計算機資源を利用したいという要求がある。例えば、プリンタはネットワーク上のどこからでも利用することができるようになっている必要がある。一方で、訪れた先の研究室などからネットワークに接続しているときに自分の研究室に設置してあるプリンタから印刷するような使い方は普通はしない。最も近くにあるプリンタを自動的に選んで印刷する機能を実現する必要がある。このような要求を満たすためには、SLP[7] のような仕組みが必要である。

分散システムが実現する透過性には様々な種類の透過性がある。情報の共有だけでなく、CPU やメモリを共有するような透過性も必要である。

1.2 FTFS(Fault Tolerant File System)

移動型計算機を構成要素に含む分散計算機利用環境を構築するためには、移動型計算機の持つ特性に適応した分散システムを構築しなければならない。

1. 計算機資源を透過的に扱うことが可能である必要がある。
2. 移動型計算機がネットワークに接続していない状態を考慮して、資源を動的に追加と削除する機能が必要。
3. 計算機資源は静的な「情報」と情報を処理する「機能」に大別できる。この二者の利用に対して、移動型計算機がネットワークと接続していない状態でも、最大限の透過性を実現する必要がある
4. 移動型計算機を含む分散計算機利用環境において、計算機資源の利用は (1) 時間的、かつ (2) 位置的に非同期に行われるので、計算機資源の状態の同期に関する問題を解決する必要がある。
5. 複数の位置固定型計算機で構成される分散計算機利用環境は一台の移動型計算機に比べて高い耐故障性を提供できる。移動型計算機に障害透過性を提供することができる。

移動型計算機は位置固定型計算機に比べてその中に保存している情報を失う可能性が高い。小型なので盗難される危険性もある。持ち運ぶので、損壊してしまう可能性もある。また、ネットワークに接続している状態でなければ、その移動型計算機内に保存してある情報を他の計算機から利用することができないのであれば、実際に損失していなくても、それが必要な瞬間には存在しないのと同じである。

本システムでは、分散環境下での移動型計算機での情報共有をリビジョン管理機能付きファイルシステムを使って実現する。FTFS(Fault Tolerant File System) では、分散環境の中でネットワーク接続性が確保できない場合を考慮し、情報共有の際に新しいリビジョンを生成し、そのリビジョンの状態を保持する。

例えば、オフィスのサーバ上のファイルを編集する場合、自動車のコンピュータに、FTFSはファイルをコピーしサーバ上では新しいリビジョンの生成をこのリビジョンの状態の管理を開始する。また、オフィスで同じファイルを編集する際には、オフィスのコンピュータに編集中の状態でない最新のリビジョンのファイルをコピーし、この新しいリビジョンを生成し状態を管理する。

リビジョン管理を徹底することにより、分散環境における情報共有の整合性の保持が実現される。このような機能は既存の分散システムでも解決しているものは少ない。本研究では、ネットワークの接続状態が予測できない移動型計算機と情報を共有するためのファイルシステムを設計し、移動型計算機を考慮した分散システムを発展させる。

第2章　移動型計算機利用環境のための分散システム

2.1　移動型計算機利用環境

移動型計算機利用環境の利用を支援する機能を実現した環境を、移動型計算機利用環境と呼ぶ。以下に移動型計算機の利用形態の具体例を示す。

- 位置固定型計算機でレポート作成やプログラムを作成する。
  ローカルで完結した作業
- ネットワークに接続してメールを受け取ったり、WWW を閲覧する。
  ネットワーク上の計算機資源を利用する
- ネットワーク上の誰かと非同期な共同作業を行う。
  コミュニケーション手段
- ネットワーク上の位置固定型計算機にログインするための端末として使う。
  利用する資源はすべてネットワークの先にある
- UPS 付き位置固定型計算機のように使う。
  小型なデスクトップマシン
- 研究室では位置固定型計算機を使い、移動中や自宅では移動型計算機を使う。
  環境に適した計算機を使う。

ネットワークが社会に広まるにつれて、移動型計算機がネットワークに接続される状況は多くなることが予想できる。しかし、移動型計算機は、その移動性が高いという特長上、駆動時間・記憶媒体・入出力デバイスなどの面でさまざまな制約がある。このため、ハードディスク容量・ネットワーク接続性などが制限されている。

位置固定型計算機によって構成された従来の環境は、ネットワーク技術の発展とともに、分散計算機利用環境として提供される場合が増えた。位置固定型計算機は、一般的には移動型計算機が持つような制約がないため移動型計算機利用環境を支援しうる資源を備えることが可能である。

移動型計算機は今後ますます増えていくことが予想できる。移動型計算機を有効に利用するためには、移動型計算機も分散計算機利用環境の一部として機能しなければならない。移動型計算機を分散計算機利用環境に統合する分散システムは、移動型計算機が位置固定型計算機や他の移動型計算機によって提供される資源の透過的な利用を可能にする機能を持たなければならない。

2.2 分散計算機利用環境の必要性

計算機システムを最も単純化してとらえると、情報とそれを処理する機能の二つから構成されていると考えられる (図 2.1)。

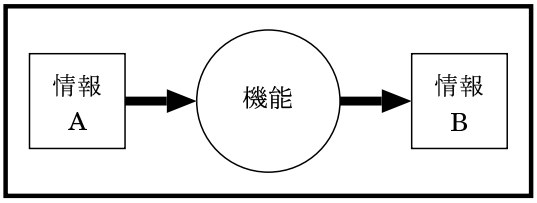

図 2.1: 計算機システム

情報とは計算処理する対象である。機能とは情報を計算処理する実体である。機能は入力情報を計算処理して出力情報を生成する。入力情報を必要としない機能もある。

計算機がたくさんあると情報や機能が分散していく。機能は本来、計算機間を移動できないが、情報はネットワークなどを介して移動することができる。現在でも、情報を移動させることにより、多くの計算機の機能を共有できるが、機能を持つ計算機上に情報をコピーしてから機能を利用し、出力情報を元の計算機上にコピーするような煩雑な操作が必要である。図 2.2 はそのような状況を表している。

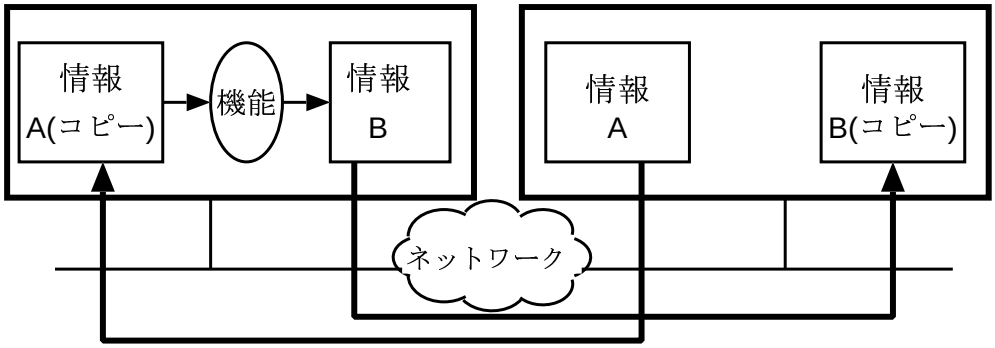

図 2.2: ネットワークシステムの情報の移動

分散計算機利用環境は、分散して存在する計算機それぞれが提供する機能や情報をネットワークを介して透過的に利用できる環境である。図 2.3 は、分散している情報や機能を共有している状態を表している。実際には異なる計算機上の資源に対する操作がネットワークを通して転送される。命令を転送するのは分散システムの役割である。分散システムがこの機能を提供し、ユーザ空間から特別な操作なしに共有が行われる。

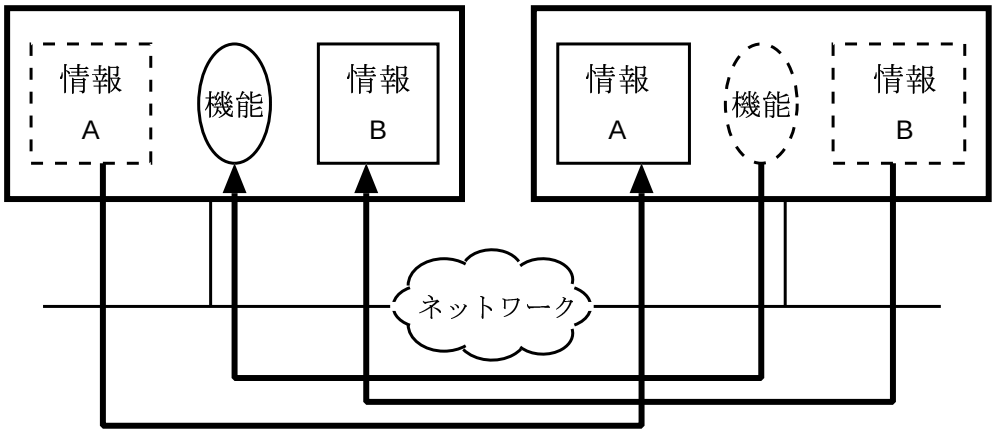

図 2.3: 分散計算機利用環境での資源の共有

分散計算機利用環境における計算機は、その多くの機能をネットワークを介して互いに共有している。

2.3 移動型計算機利用環境の必要性

分散計算機利用環境はネットワーク上にある分散した計算機を統合した環境である。ネットワークに接続していない計算機を統合することは不可能である。移動型計算機はその移動するという特徴のため、ネットワークへの接続性を常に確保し続けることが困難である。ネットワークを介した資源の共有を実現する移動型計算機利用環境において移動型計算機を他の移動しない計算機資源と同様に扱っていくことは難しい。

また、移動型計算機がネットワークに接続する位置は変わる可能性がある。ネットワーク上で移動型計算機を表す識別子も変わる可能性がある。移動した先が管理ドメインの異なるネットワーク上であることも考えられる。適切な計算機資源を利用する機能を実現する必要がある。

現状では、このような移動型計算機のふるまいに対して分散計算機利用環境を提供するための方法がない。このため移動型計算機上では図 2.2 で示した利用モデルでしか利用できない。

しかし、機能や資源には制約のある移動型計算機でこそ、ネットワーク上に散在する計算機資源を有効に利用できるべきである。移動型計算機利用環境では移動型計算機の利用を支援する機能が実現されなくてはならない。

2.4　分散システムでの解決

分散システムは計算機の利用者に対して、最大限の利便性をもたらすことを最終的な目的にしているシステムである。したがって、分散システムによって解決しようとしている問題は広い範囲にわたっている。分散システムによって解決しようとしている問題の具体例を示す。

高性能性

非常に高い計算能力を実現するためにはハードウェアの高性能化だけでは不十分である。高い計算能力をもつハードウェアを作ることには技術的な限界や、物理的な限界があるからである。このようなハードウェア開発の限界をソフトウェア的に解決することが考えられる。具体的には一つの処理を複数の計算機上に分散させ、並列計算することが考えられる。

操作性の統一

計算機には様々な種類が存在している。それぞれが特徴を持っていてその全てを網羅するような計算機は存在しない。様々な計算機は独自のインターフェースを備えていると考えられる。しかし、このような異機種性は利用者に負担になる。表面的な計算機のインターフェースを統一して、内部でそれぞれの目的や処理にあった計算機を自動的に選択することが考えられる。

規模拡張性

柔軟な規模拡張性。既存のシステムを高性能なシステムで置き換えるのではなく、既存のシステムに必要最小限の機器の増設によって必要な性能を得られるような規模拡張性を実現できる。

障害発生の確率を低くする

計算機は様々な要因で処理を中断しなければならない状態に陥る。このような障害は、計算機の応用分野によっては重大な問題を引き起こす。一つの処理を冗長な処理で実現することによって、なんらかの障害から自動的に復旧することや、

障害の発生を隠蔽することが期待できる。

効率的な計算機の利用

一つの処理を一つの計算機で行うことは、負荷の集中を引き起こす。表面的には一つの処理を、複数の計算機で実行して負荷分散を実現することができる。負荷の集中をプログラムの実行中に検出して、より負荷の低い計算機にプロセスを移送するような解決方法も考えられる。

ユービキタスコンピューティングを実現する

利用者は様々な場所から計算機を利用したいという要望を持っている。異なる計算機上に常に同じ環境を構築する必要がある。

計算機の管理コストを低減する

多くの計算機を管理する場合、その管理コストは計算機の数に比例して、またはそれ以上に増大する。特に人的な管理コストを低減するためには、計算機の管理者の負担を減らす必要がある。具体的には、ネットワークコンピュータのようにソフトウェア的な管理対象を集中させ、処理だけを分散させるような解決方法が考えられる。

管理ドメインを越えた分散計算機利用環境の構築を可能にする

管理ドメインとは、計算機システム運用の一つの規定が適用される範囲のことである。このドメインの運用は管理者の権限を与えられた一人または数人の人間によって実行される。管理ドメインが適用される範囲は社会的な理由によって決まる。多くの場合、所有者や管理者、もしくは利用する権利を持つユーザの集合が同じである計算機に対して一つの管理ドメインが存在する。具体的な例を挙げると、大学にある計算機はユーザの集合によって管理ドメインが決まっている。さらに、大学が用意している計算機とは別に、研究室で利用している計算機はユーザを研究室に所属している人に限定するために、異なる管理ドメインになっていることが考えられる。最後に、個人で所有している一台計算機も、他の管理ドメインには入らない。

管理ドメインの境界は、計算機間を接続する際の技術的な境界ではない。したがって、多くの既存の分散システムでは管理ドメインの境界に対する透過性を考慮していない。この境界を越えて、計算機資源へのアクセスに対する透過性を実現可能な分散システムは存在するが、位置透過性を実現することは困難である。

その原因は、計算機上でユーザを表す識別子は管理ドメインの範囲内において

一意であることを保証するからである。異なる管理ドメインの間でそれぞれの管理ドメイン上のユーザを表す識別子を対応させる方法が、実現されなくてはならない。

その計算機群を利用するユーザの集合計算機を利用する権限をもつ利用者の集合によって決まる。

2.5 分散システムが提供するべき機能

以上のような様々な問題を解決するために必要な機能をまとめると以下のようになる。

- 位置透過性
- 障害透過性
- 並列性透過性
- 負荷分散
- 異機種透過性
- セキュリティ
- 規模拡張性
- 操作性の統一

これらの問題を解決する機能をあわせ持ったシステムを実現することが、移動型計算機利用環境を実現することである。

2.6 分散システムの実現方法

分散システムの実現方法は、以下の四つに分類することができる。

- 分散オペレーティングシステムを作る方法
- 仮想計算機を作る方法
- アプリケーション毎に解決する方法
- 拡張システムを追加する方法

それぞれの特徴について以下で論ずる。

2.6.1　分散オペレーティングシステムでの実現

既存のシステムは分散システムとは設計が根本的に異なるために、既に述べた様々な問題を解決するために必要な機能を実現することは困難である。したがって分散システムに求められているより多くの問題を解決しようとするならば、既存のシステムとは異なる新たなシステムを作る必要がある。

分散オペレーティングシステムを作る方法は新たなシステムを作る方法の具体的な手段である。オペレーティングシステムは計算機ハードウェアをソフトウェア的な計算機資源に変換する役割を担っている。多くのプログラムはオペレーティングシステムの支援を受けて動作する。分散オペレーティングシステムはプログラムが動作するための基盤環境に分散システムとしての機能を実現することが可能である。

分散オペレーティングシステムの長所は、分散システムに求められる要求の全てを実現できることである。

分散オペレーティングシステムによる方法の短所もある。ひとつはオペレーティングシステムの開発には時間がかかることである。また、既存のソフトウェア資産を生かせないことも問題である。さらに、既存の全てのオペレーティングシステムを新しい分散オペレーティングシステムで置き換えなければならない。現実的にはこれは不可能である。

2.6.2　仮想計算機での実現

仮想計算機を作る方法は、分散オペレーティングシステムを作る方法に似ている。仮想計算機は既存のシステム上にソフトウェア的に新たな計算機を実現する。したがって、新たな仮想計算機上では分散システムで解決しようとしている問題の全てを実現することができる。仮想計算機による方法は、分散オペレーティングシステムと違って全ての既存のシステム上で動かすことが可能である。既存のソフトウェア資産を分散システムの中で使うことはできないが、新たに作成する分散システムと同時に利用することができる。このことは利用できるという長所である半面、分散システムとは異なる空間であるために、より高い透過性を提供できない点で短所でもある。

2.6.3　アプリケーション毎の実現

分散オペレーティングシステムによる解決方法と最も発想が異なる解決方法はアプリケーション毎に解決する方法である。利用者が利用するアプリケーションの数は非常に多い。このため非常に多くの変更を行わなければならない。実現方法が困難であるにも関わらず、分散システムによって解決しようとしている問題のうち、いくつかの限られた問題だけが解決できる。また、改変できるソフトウェ

アも限られるために有効な手段ではない。

2.6.4　拡張システムによる実現

拡張システムを追加する方法は他の 3 つの方法の中間に位置する。分散オペレーティングシステムや仮想計算機で分散システムを実現するのに比べれば制約は多い。一方、アプリケーション毎に解決する方法に比べれば、制約も少なく既存のアプリケーションの全てが、提供される分散システムの機能を享受できる。

拡張システムで実現する方法の最も重要だと考える長所は、既存のソフトウェア資産をそのまま利用できる点である。また、既存のシステムとの親和性が高いことは、システムの移行コストが極めて低いということであり、新たにシステムを実現する際の重要な点と言える。

2.7　計算機資源の分類

分散システムを実現するにあたって、計算機資源の分類を行う必要がある。

計算機資源は計算機システム上の全ての資源の総称である。実際には計算機資源はふるまいや性質が異なるさまざまな種類の資源が含まれている。

ふるまいや性質が異なる計算機資源を、分散システムは明確に分けて、それぞれに対して最適な分散計算機利用環境上での適切なふるまいを提供しなければならない。

この視点にしたがって、計算機資源は大きく二つに分類できる。静的な情報を保持する計算機資源とそれを処理する動的な計算機資源である。後者の計算機資源は、異なる種類の計算機で分散計算機利用環境を構築する分散システムにおいて、複製や移動などが前者に比べてより難しい。

計算機資源はまた機能資源と情報資源だとも言える。情報資源は静的な計算機資源であり、機能資源は動的な計算機資源であると。例えば、処理にかかわる演算装置、一次記憶装置、実行中のプログラムの実行体といったものが機能資源であり、入出力にかかわるデータ、二次記憶装置、出力デバイスといったものが情報資源である。

機能は実行環境に依存しているため、移動することが難しい。機能は、情報を処理するための命令の列を実行環境である計算機システムに与える役割と、情報を処理する間の計算機システムの状態を保持する役割を持っている。実行環境と密接な関係にあるため、移動するのは困難である。

一方、情報は異なる計算機システムごとにエンコーディング方式の違いはあるが、それらの間での交換は可能である。情報を複製することも可能である。したがって、機能の移動や複製を行う場合と比較して、情報の移動や複製は容易である。

2.7.1 静的な計算機資源

静的な計算機資源の具体的な例は、文書ファイルやプログラムのソースコードなどである。静的な計算機資源は情報を保存するために使用される。記録されている情報は、ユーザによって明示的に変更しない限り変化しない。

静的な計算機資源に対して様々な透過性を付加することは、分散システムの基盤を構築することである。静的な計算機資源は動的な計算機資源によって処理される。動的な計算機資源の移動や複製には制約が多い。したがって、計算機資源の移動や複製という操作を、静的な計算機資源を移動や複製することによって疑似的に実現する方法が考えられる。したがって、静的な計算機資源に対する透過性の実現は、動的な計算機資源に対する透過性の実現を左右する可能性がある。

静的な計算機資源に対する、分散システム上での操作の意味づけを以下に示す。

- 共有
  アクセス透過性の実現。分散計算機利用環境上の計算機が計算機資源の位置を考えずに利用できること。

- 複製
  耐故障性の提供。誤動作で失っても別の複製をもとに再生できる。

- 移動
  性能向上のため。ネットワーク越しに利用するのに比べ、データを移動して処理するほうが、処理にかかる時間を短縮できる場合がある。

2.7.2 動的な計算機資源

動的な計算機資源の具体的な例は、実行中のワードプロセッサアプリケーションがこれにあたる。実行中のプログラムは、実行を支援しているオペレーティングシステムと密接に結び付いている。このため移動は実行を支援しているオペレーティングシステムを変更しなければならない。物理的な資源に直接結び付いた計算機資源は移動することも複製することも不可能である。

したがって、共有や移動や複製の要求に対して異なる意味を与えるか、もしくは仮想的にそれらを実現するための仕組みを、分散システムは提供しなければならない。

これらの計算機資源は、ユーザの操作とは非同期に内部状態が変化する。

動的な計算機資源に対する、分散システム上での操作の意味づけを以下に示す。

- 共有
  プロセスの遠隔実行。

- 複製
  耐故障性の実現。処理結果を比較して処理の正しさの信頼度をあげる

- 移動
  負荷分散。空いている計算機資源の有効利用。ネットワーク切断時にも計算機資源の継続利用を可能にする。

第3章　移動型計算機利用環境を考慮した分散システムの設計

3.1　分散環境モデル

以下に理想の分散システムの基本的なモデルを述べる。

3.1.1　仮想計算機

分散している複数の計算機を分散システムによって一つの仮想計算機に統合する。それぞれの分散している計算機を、仮想計算機を構成するノードと呼ぶ。それぞれのノード上の計算機資源を仮想計算機上の資源として透過的に利用できる。の物理的な位置に関わらず透過的に利用できる。

位置固定型計算機だけでなく、移動型計算機も仮想計算機を構成するノードになりうる。それぞれのノードは所属する管理ドメインが異なっていても構わない。

ユーザによって利用可能な計算機の集合は異なる。したがって、仮想計算機は一人のユーザに一つづつ作られる。あるユーザの利用可能な計算機の集合で構成される仮想計算機は、そのユーザが所有している仮想計算機と言う。

分散計算機利用環境には利用者と仮想計算機、位置固定型計算機、移動型計算機、管理ドメインという要素がある。それぞれのの関係を図 3.1 に示す。

3.1.2　ユーザの分類

仮想計算機上には、その所有者以外のユーザのアカウントを作成できる。所有者以外のアカウントは、所有者の利用可能な資源の共有のために作成する。このアカウントを作成する権利は所有者だけに与えられている。この権利を所有者以外に委譲することはできない。

仮想計算機上の利用者は三つのクラスに分類される。

- 所有者
- グループ
- 所有者ではなく、グループにも所属しない

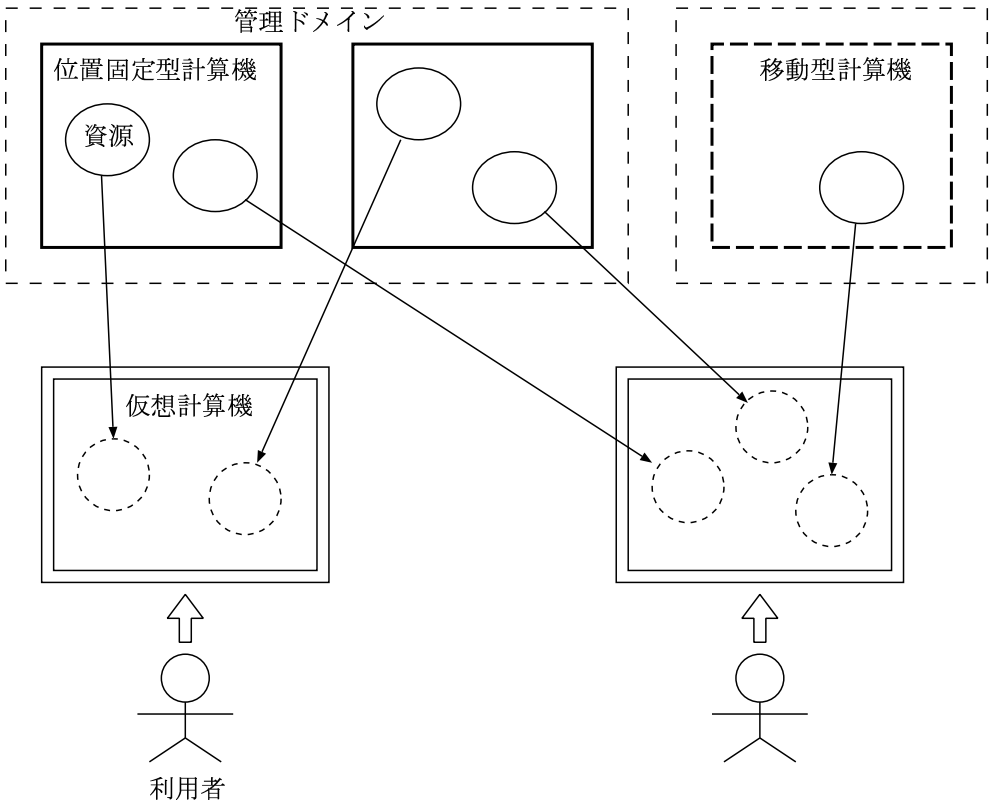

図 3.1: 移動型計算機を考慮した分散計算機利用環境のモデル図

仮想計算機上では、複数のユーザをグループにまとめることができる。ユーザは複数のグループに所属することができる。グループは仮想計算機の所有者によって作成される。

3.1.3 資源の属性

仮想計算機上の全ての資源は属性値を持つ。属性値は次のように分類できる。

- アクセス権を示す属性
- 所有者を示す属性
- 資源の物理的な位置を示す属性

アクセス権

資源はアクセス権を示す属性を持つ。この属性は資源の所有者だけが変更する権利を持つ。アクセス権を示す属性は、ユーザの分類である三つのクラス毎に異なる値を持つことができる。属性の値は次の三つの属性値の任意の組み合わせである。

- 資源の内容を読むことを許可する
- 資源の内容を書き換えることを許可する
- 資源を実行することを許可する

所有者

仮想計算機上の資源は所有者を表す属性を持つ。仮想計算機上にある資源の所有者を変更することはできない。

仮想計算機上にある資源のうち所有権が所有者にない資源は、指定された属性に関わらず、仮想計算機上の他のユーザと共有することはできない。これは資源のまた貸しを制限するためである。

位置情報

仮想計算機上の資源は実際には分散して存在している。

移動型計算機がネットワーク切断状態になった時に、資源が移動型計算機上に複製または移動されていなければ利用することができない。また、利用可能な計

算機資源の総量は計算機毎に異なるので、全ての資源を全ての計算機上に複製することもできない。

資源を利用する際には位置情報を知る必要はない。しかし、資源が物理的にはどこにあるかを調べる手段は提供する必要がある。

この属性は移動型計算機を支援する機能にもなる。

位置情報は資源が実際に存在するノードの名前のリストである。信頼性を提供するために資源は複製される可能性がある。

時間情報

作成された時刻、最後に変更を行った時刻、最後に参照した時刻。全ての資源はこの三つの時間の情報を持つ。複数のバージョンを持つ資源、最新のものを決定するためにも重要である。

バージョン情報

ノードに移動型計算機が含まれている。移動型計算機はネットワークに接続されていない状態がある。この状態で計算機資源を変更すると、ネットワーク上の複製との一貫性がなくなる。資源は非同期に変更することを禁止すれば問題は解決できるが、この方法は望ましくない。

全ての資源はバージョン管理される。非同期に修正された資源は異なるバージョンでシステム上に保存される。

普通は最も新しい時間の属性情報を持つものが、複数のバージョンがある資源の最新の内容として利用される。

しかし、一貫性を失った資源は最新の資源の内容で上書きするのではなくて、自動的に違うバージョンとして同じ名前で管理される。

3.2　既存のシステムとの親和性

分散システムを実現する手段はいくつか存在する。既存のシステムに変更を加えたり、機能追加するプログラムを実行するによって分散システムを実現する方法が、もっとも現実的である。

ユーザが利用する権利を持つ全ての計算機を一つの分散計算機利用環境環境として利用することが目的である。管理ドメインが異なる全ての計算機に新しいシステムをインストールすることは不可能である。したがって、新しい分散オペレーティングシステムを作る方法では目的を達成できない。

既存のシステムとの親和性は、分散システムにとって重要な機能である。既存の分散オペレーティングシステムは、分散計算機利用環境を構成するノードをそ

の分散オペレーティングシステムを利用する計算機に限定している。現実的には全ての計算機が一つの分散オペレーティングシステムによって動作するような状況はあり得ない。

したがって、分散オペレーティングシステムは既存のシステムを分散計算機利用環境に統合するための何らかの手段を実現する必要がある。そうしなければ、異なる種類のオペレーティングシステムを実行している計算機に対しては透過性を実現できないので、ユーザにとって計算機利用環境は分散して存在することと変わらなくなってしまう。

管理ドメインが異なる計算機のことを考えると、分散システムの実現には管理者の権限を必要としないことが望ましい。

3.3 設計

3.3.1 分散システムの機能を実現する機能部分の分類

分散システムは様々な機能を実現する必要がある。それらの機能を性質や対象によって分離して複数の機能部分の集合によって一つの分散システムを実現する。

まず分散システムを大きく二つに分類する。

- 情報資源を透過的に扱う機能部分
- 機能資源を透過的に扱う機能部分

二つに分離した理由は、情報資源と機能資源には同じ透過性の概念に対して異なる意味を持つからである。

この二つの資源はさらにそれぞれが以下の三つの機能部分が協調して動作することによって実現する。

3.3.2 情報資源を透過的に扱うための機能部分

- 名前管理部分
- 資源提供部分
- 資源利用部分

名前管理部分

名前管理部分は分散して存在する各ノード上の資源の名前と分散計算機利用環境上での名前との対応を管理する。名前の解決はこの部分によって実現する。

名前の解決は分散システムの最も重要な機能である。名前が解決できなければ、他の機能部分が動作していたとしても、資源の透過的なアクセスは不可能である。また、移動型計算機にはネットワーク切断状態があるので、移動型計算機上に名前管理部分がなければならない。したがって、名前管理部分は複数のノード上で動作している。

名前空間の一貫性を保証するために各ノード上の名前管理部分は協調して動作する。

名前管理部分は資源利用部分から、名前の解決を依頼される。この依頼に応じて名前を解決する。指定された資源を供給する資源提供部分に、資源の供給命令を発行する。

名前管理部はシステム全体を管理する役割も担っている。ユーザに透過的な資源の移動や、複製、バージョン管理による一貫性の保証などの機能を実現する。

資源提供部分

資源提供部は、ノード上の資源アクセスをネットワークの先からの要求に基づいて代行する。名前管理部から資源提供命令を受け取り、名前管理部に指定された資源利用部分に対して資源利用の代行サービスを提供する。

全ての資源へのアクセスは、資源提供部によって実際の資源へのアクセスに変換される。

資源利用部分

既存のプログラムは、ノード上のローカルな資源にアクセスするよう記述されている。したがって、プログラムが直接、資源提供部分が提供する資源アクセスサービスを利用することはできない。したがって、資源利用部分が資源アクセスサービスとの通信を行う。

資源利用部分は既存のプログラムからの資源アクセス要求を受け取るために、オペレーティングシステムの一部として動作しなければならない。情報資源はファイルとして提供する。したがって、資源利用部分はファイルシステムのインターフェースを持つ。

今日稼働している多くのオペレーティングシステムは、NFSのようなネットワークファイルシステムをマウントする機能を持つ。したがって、オペレーティングシステムを直接変更せずに拡張することができる。例えば、資源利用部分はオペレーティングシステムに対してはNFSと同じインターフェースを提供することによって実現できる。

資源に対する透過的なアクセス以外の機能は、そのような機能が存在しないシステム上のプログラムから利用できない場合もある。例えば、ある資源に古いバージョンが残されているとすると、既存のプログラムからそのバージョンにアクセ

スすることはできない。したがって、新たな機能を利用するためにファイルシステムインターフェース以外のインターフェースを提供しなければならない。
またこのインターフェースを利用するプログラムを作成するためのライブラリも提供しなければならない。

3.3.3 機能資源を透過的に扱うための機能部分

基本的には、機能資源を透過的に扱うための機能部分も、情報資源の場合と同じように三つの部分で構成される。

- 名前管理部分
- 資源提供部分
- 資源利用部分

ユーザが分散システムを管理する際の利便性のために、システムを可能な限り単純な構成にするべきである。この目的のために、機能資源を扱う三つの部分は情報資源での三つの部分と同じモジュール内で実現する。
それぞれの管理部分は、ユーザに指定された資源が、情報資源か機能資源かを判別する。その後それぞれにふさわしい機能を提供する。
図 3.2 に上記の三つの機能部分の関係を示す。

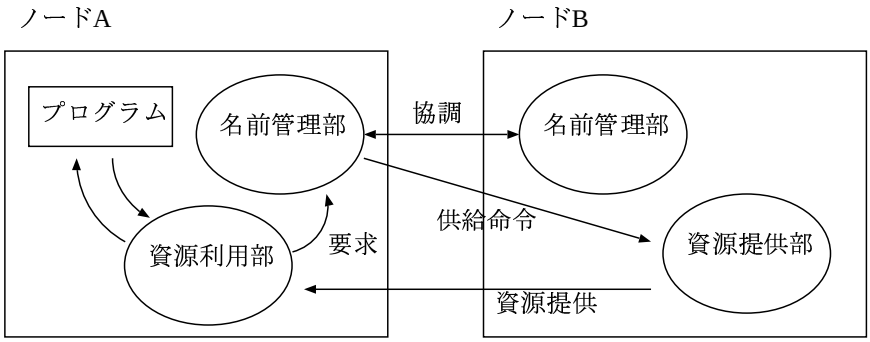

図 3.2: 各機能部分の関係

3.4 分散ファイルシステム

分散ファイルシステムは情報資源に対する透過性を提供する機能部分に相当する。全ての情報資源はファイルとして扱う。

本論文では上述した分散計算機利用環境を実現するために、情報資源を透過的に扱うための機能部分である分散ファイルシステムを設計する。

このファイルシステムは移動型計算機が分散計算機利用環境の一部であることを前提に、情報資源の共有の際の課題を明確にし、それを克服している。

特徴的なのは、資源の非同期な操作にたいする一貫性をバージョン管理という方法で実現していることである。移動型計算機のネットワーク切断性による、情報資源への非同期な操作が行われる可能性がある。しかし、資源の一貫性を保つために、非同期なアクセスを制限するようなことは、利便性を損ねる。

そこで、バージョン管理機能をファイルシステムレベルで実現している。この機能によって移動型計算機にとっては、非同期な資源アクセスが可能になる。また、位置固定型計算機にとっては、人間の誤操作やシステムの誤動作による資源の破損や紛失に対する信頼性を向上する機能を実現できる。

第4章　関連研究

4.1　関連研究について

- 異機種混合環境、ヘテロジーニアスな環境をサポートしているかどうか。
- 分散システムの機能のうちどの機能を実現しているのか。なにが実現されていないのか。
- 分散システムの一部がネットワーク接断した時の振舞。
- ネットワーク切断透過性の実現。
- 環境構築する権限をもつのは管理者か利用者か。

4.2　NFS

NFS(図 4.1) は、ほとんどのオペレーティングシステムにおいて実装されており、現在最も広く利用されるネットワーク共有ファイルシステムである。位置固定型計算機間でのファイルシステムの共有を想定している。

NFS は、ネットワークを介して利用可能なファイルシステムを、ローカルの計算機資源と同様のアクセス方法で扱うことを実現している。アプリケーションからの NFS へのアクセスをローカルのファイルシステムへのアクセスと全く同様に行なえるように、NFS はオペレーティングシステムの内部に実装されている。利用者はアクセスするファイルがローカルのファイルシステム上にあるのか、あるいはネットワークファイルシステム上にあるのか意識する必要がないため、ファイルの位置透過性が実現されていると言える。

NFS は、計算機の停止やネットワークの切断からの復旧時に必要とされる処理を最小限に留めるため、状態を持たないサーバ、クライアント型のシステムとして実現されている。サーバ、クライアント双方でキャッシュを持つことが可能だが、NFS におけるキャッシュの目的は効率の向上であり、キャッシュの有効期限は秒単位のとても短かいものに設定される。移動型計算機を考慮しないため、ネットワークの切断時にはこのキャッシュは利用できず、NFS 上のファイルを操作することはできない。また、複数の利用者が同時に一つのファイルを操作することは考慮されておらず、後に操作されたものがシステムの状態として保存される。

NFS のファイル空間は、クライアントとなる計算機のローカルファイル空間の一部として利用する。ローカルファイル空間のどこに NFS のファイル空間を配置するかは、クライアントとなる計算機の設定によって変更できる。

NFS サーバは、すでに NFS を利用して実現しているファイル空間を NFS クライアントに提供するなど、あるファイルに対するアクセスが二度 NFS によって処理されるような利用を禁止している。なぜならば、禁止しない場合、NFS が持つネットワークアドレスによるアクセス制限の機能に意味がなくなってしまうからである。

図 4.1 に NFS の概念図を示す。

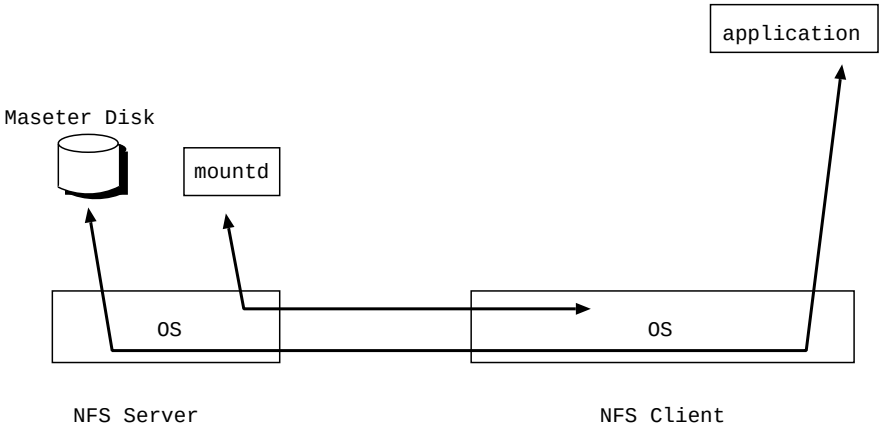

図 4.1: NFS の概念図

NFS の特徴を以下にまとめる。

- 複数の利用者が同一のファイルを操作することを考慮しない
- ネットワーク環境の変化に対応しない
- NFS の多段利用はできない

本研究の視点から見た NFS の特徴を以下に述べる。

- 異機種混合環境をサポートしている
- 情報資源を透過的に扱うための名前管理部分を持つ
- 情報資源を透過的に扱うための資源提供部分を持つ
- 情報資源を透過的に扱うための資源利用部分を持つ
- 機能資源を透過的に扱うための名前管理部分を持たない

- 機能資源を透過的に扱うための資源提供部分を持たない
- 機能資源を透過的に扱うための資源利用部分を持たない
- ネットワーク切断透過性を実現しない
- 利用者は環境構築する権限を持たない

4.3 Personal File System

Personal File System(以下 PFS と書く)[6] は、移動型計算機において、他の計算機が持つ情報を効率的に利用するための研究である。ここで、他の計算機とは位置固定型計算機を想定している。移動型計算機における位置固定型計算機上の情報の利用は、ファイルシステムを共有することで実現される。

移動型計算機で共有ファイルシステムを利用するアプリケーションの効率は、ネットワークの状態によって大きく影響を受ける。PFS は、ネットワークが切断されている状態や、低速なネットワークを利用している場合においても、アプリケーションの作業効率を低下させないことを目的としている。

これを実現するため、PFS はサーバ、クライアント型のシステムを構築する。位置固定型計算機をファイルサーバとして利用し、クライアントとなる移動型計算機ではファイルのキャッシュを持つ。ネットワークの利用が制限される場合にはキャッシュ上のファイルを用いてアプリケーション作業を続行する。移動型計算機のネットワーク利用が可能になった時点で、CUP(Cache Update Protocol) と呼ばれる独自のプロトコルを用いて、位置固定型計算機上のサーバと移動型計算機上で変更されたファイルの同期がとられる。位置固定型計算機と移動型計算機上の同一のファイルがそれぞれ非同期に変更された場合、その旨を利用者に通知するのみで、同期をおこなわない。このため、PFS はファイルの一貫性を保証しない。

ファイルサーバ上では MS(Master Server)、クライアント上では CS(Cache Server) と呼ばれるユーザプロセスが動作する。MS は位置固定型計算機上のオペレーティングシステムが提供する入出力機能を用いてマスターのファイルを管理する。また、MS は前述の CUP を用いて CS からのファイル入出力要求を受け付ける。CS は CUP を用いて入手したファイルのキャッシュを移動型計算機上で管理すると共に、移動型計算機のオペレーティングシステムに対して NFS サーバの役割を果たす。この様にして、PFS は次の二つのことを実現している。

- オペレーティングシステムや既存のアプリケーションを変更しないので、導入が容易であること
- 既存のファイルシステムとの共存および PFS システム自体の透過性の実現

図 4.2 に PFS の概念図を示す。

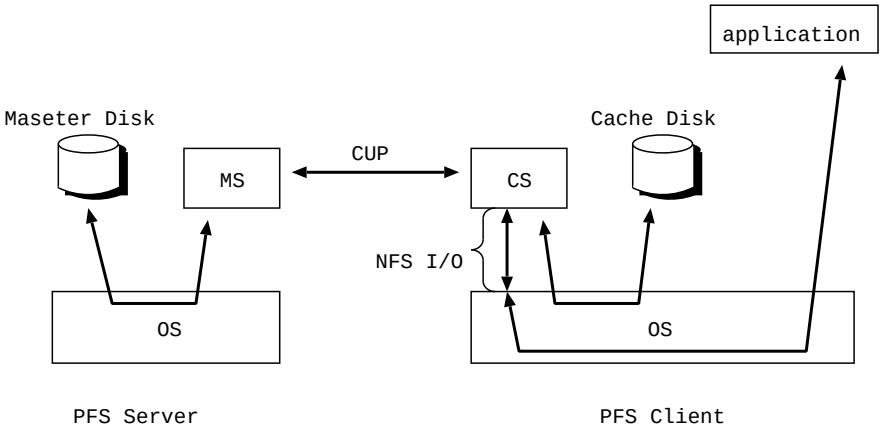

図 4.2: PFS の概念図

また、PFS の利用者は動作モードを設定することによって、同期方法をおおまかに指定することができる。

- NFS 互換モード

  NFS のようにファイルアクセスと同期してキャッシュの更新を行うモード。高速なネットワークの利用を想定しており、ファイルの一貫性が保証される。低速なネットワークの利用時やネットワーク切断時には適さない。

- 非同期更新モード

  キャッシュに存在するファイルの入出力を行なう際に、サーバからの応答を待たずにキャッシュ上のファイルを利用するモード。定期的にキャッシュのサーバファイルとの一貫性がチェックされ、変更されたファイルは可能な限り速やかにサーバに反映される。低速なネットワークの利用時もファイルの入出力にかかる時間は少ないが、ファイルの一貫性保持が不完全である。

- 読み込み専用モード

  サーバからの読み込みによるキャッシュの更新だけを行ない、ファイルの変更をサーバに書き戻さない。非同期モードと同じように定期的にキャッシュの一貫性がチェックされるが、サーバに書き戻さないため通信量を削減することができる。書き戻しが許される他のモードに遷移するまでは、クライアント上でのファイルの変更はマスターファイルに反映されない。

- 切断モード

サーバとの通信は一切行なわれない。キャッシュに存在しないファイルへの入出力は実行されず、エラーとなる。読み込み専用モードと同じく、ファイルの変更は書き戻しが許される他のモードに遷移するまでサーバのマスターファイルには反映されない。

以下に PFS の特徴をまとめる。

- システムがユーザ空間のプロセスによって実現されている
  - 導入が容易である
  - 既存の技術と組み合わせた利用が可能である
  - ユーザ空間とカーネルの間の処理が増えるため、オーバヘッドが大きい
  - カーネルの持つ機能を越えたシステムとして拡張することが出来ない
- 移動型計算機から位置固定型計算機上の情報を共有することが目的なので、位置固定型計算機から移動型計算機の情報へのアクセスは考慮していない

本研究の視点から見た NFS の特徴を以下に述べる。

- 異機種混合環境をサポートしている
- 情報資源を透過的に扱うための名前管理部分を持つ
- 情報資源を透過的に扱うための資源提供部分を持つ
- 情報資源を透過的に扱うための資源利用部分を持つ
- 機能資源を透過的に扱うための名前管理部分を持たない
- 機能資源を透過的に扱うための資源提供部分を持たない
- 機能資源を透過的に扱うための資源利用部分を持たない
- ネットワーク切断透過性を実現する
- 利用者は環境構築する権限を持たない

4.4 Amoeba

Amoeba[10] は、並列処理を考慮した分散オペレーティングシステムである。今までのオペレーティングシステムにあった概念や制約を受け継がず、分散システムに求められる新しい概念の構築とその実現を行なっている。

Amoeba は単一または複数の計算機をグループ化したものの上で動く一つのシステムである。システム全体で一つのオペレーティングシステムの役割を果たす。

Amoeba では全ての計算機資源は”オブジェクト”として表現される。オブジェクトは、抽象型データ型であり、その内部には隠蔽されたデータとそれに対する操作が格納されている。これらのオブジェクトの位置を示すものとして”ケーパビリティ”という識別子がある。これは Amoeba システム内で一意である。

Amoeba によって管理されるすべての計算機では、それぞれ”Amoeba マイクロカーネル”が動作している。これは以下を含む。

1. プロセスとスレッドの管理機能

   Amoeba では、UNIX や MS/DOS に似たプロセスという概念を持つ。さらに、プロセス内では単一または複数のスレッドが動作する。プロセスはそれぞれ固有のアドレス空間、レジスタ群、スタック等を持つ。その内部で動作するスレッド群は、さらに固有のレジスタ群、スタック等を持つ。同一プロセス内で動作する複数のスレッドは、同一のアドレス空間を使用する。

2. 計算機における各装置のドライバ

   Amoeba マイクロカーネルは、計算機に付随する個々の入出力装置に対応するデバイスドライバを持つ。これらはマイクロカーネルとリンクされており、動的にロードすることはできない。

3. メモリ管理の支援機能

   プロセスが動作するためにマイクロカーネルは、一つ以上の”セグメント”と呼ばれるメモリのかたまりをプロセスに割り当てる。

4. 通信の支援機能

   マイクロカーネルは、最も基本的なプロセス間通信を提供する。提供する通信は 2 地点間通信とグループ間通信である。

ここで、例えばメモリセグメント、プロセッサ、プロセスやファイルもオブジェクトとして表現され、ケーパビリティによって指定される。

マイクロカーネルが処理しないものはすべてサーバプロセスが処理する。オブジェクトはサーバが管理する。プロセスがファイルオブジェクトを生成し、その識別子であるケーパビリティを取得する処理過程は以下になる。

1. クライアントプロセスがスタブ手続きを用いて、サーバにファイルオブジェクトの生成を依頼する
2. ファイルサーバはファイルオブジェクトを生成し、オブジェクトとケーパビリティの対応を表に記録する。クライアントプロセスにはケーパビリティが返される。

オブジェクトは、Amoebaシステム内で一意に識別される。このため、ファイルシステムの共有はケーパビリティを用いることで実現できる。ファイルのパス名とケーパビリティの対応は、ディレクトリサーバによって行われる。ファイルにアクセスするとき、プロセスはディレクトリサーバにパス名の探索を要求する。探索が成功した場合にはケーパビリティが返され、プロセスはそれを用いてファイルサーバにアクセスする。

処理の分散の実現は、プロセス毎に行なわれる。プロセスの実行時には、引数にプロセスサーバのケーパビリティとプロセス記述子を設定したスタブ手続きが呼ばれ、プロセスサーバがexec関数を用いてプロセスを実行する。

本研究の視点から見たAmoebaの特徴を以下に述べる。

- 異機種混合環境をサポートしている
- 情報資源を透過的に扱うための名前管理部分を持つ
- 情報資源を透過的に扱うための資源提供部分を持つ
- 情報資源を透過的に扱うための資源利用部分を持つ
- 機能資源を透過的に扱うための名前管理部分を持つ
- 機能資源を透過的に扱うための資源提供部分を持つ
- 機能資源を透過的に扱うための資源利用部分を持つ
- ネットワーク切断透過性を実現しない
- 利用者も環境構築する権限を持つ

第5章　Fault Tolerant File System

5.1　目的

Fault Tolerant File System(以下 FTFS と書く) は、移動型計算機を構成要素に含む分散計算機利用環境 を実現するための分散システムの、計算機資源への透過的なアクセスを実現する部分だけを取り出したものである。計算機資源の透過的なアクセスは分散システムの基盤になる部分で、その機能や性質によって分散システムが実現する機能に制約をあたえる。

FTFS は分散システムの構築が最終的な目的である。FTFS を基盤にした分散システムを想定し、その分散システムを実現可能にするための機能を FTFS で実現する。

5.1.1　FTFSが実現するべき機能や制限

FTFS は分散計算機利用環境に移動型計算機を含むことを前提にした分散システムの一部である。移動型計算機 を含めるために必要な事柄を含めて、FTFSが実現するべき機能や制限を列挙し、後に具体的に説明する。

- 位置透過性
  - 管理ドメインの境界を越えて一つの名前空間を提供する
  - 名前には計算機資源の位置を示す情報を使わない
  - FTFSを利用した分散システムが負荷分散やプロセス移送などを行えるようにする
  - 管理ドメインの境界を越えられること
- 障害透過性
  - 計算機資源を自動的に複製して、誤動作や誤操作による損壊、紛失を防ぐ
- 高性能性
  - ネットワークの帯域が狭い場合でも、適切な早さで計算機資源を利用できる

  - FTFSを利用した分散システムが負荷分散やプロセス移送などを行えるようにする
  - 計算機資源の名前解決を速く実行できること
- 並列性透過性
  - 計算機資源が非同期に扱われる際の一貫性の保証
- 操作性の統一
  - 種類が異なる計算機システムの間で適切な操作を行えること。主に計算機資源の属性の扱い方について統一されたインターフェースを提供すること
- 規模拡張性
  - 利用できる計算機資源の量は分散計算機利用環境を構成する計算機の数に比例する。
- 障害発生の確率を低くする
- 効率的な計算機の利用
  - FTFSを利用した分散システムが負荷分散やプロセス移送などを行えるようにする
- セキュリティ
  - 管理ドメインを越える分散計算機利用環境の構築は、ユーザの所有している計算機資源だけでなく、計算機システム全体を危険にさらす可能性がある。この危険性を排除する。
- 計算機の管理コストを低減する
  - 実行時の環境の変化に自動的に適応できること
  - 単純な概念の組み合わせで実現する
- ネットワーク切断状態に対する透過性
  - ネットワークの切断時にも可能な範囲で計算機資源の共有を実現する
  - ネットワーク切断時に分散計算機利用環境の共有空間にある計算機資源を非同期に扱う
  - 移動型計算機がネットワーク切断状態であっても、それが原因でシステム全体が計算機資源や不正な状態に陥らないこと

FTFSが満たすべき要求のリスト。

- 計算機資源のアクセスや利用に対する位置透過性を実現すること
- 移動型計算機のネットワーク切断に対する透過性を実現すること
- ユーザに利用許可を与えられている情報は、ユーザが利用する全ての計算機上で同じ名前でアクセスできること
- 情報を損壊するような障害を回避する障害透過性の実現
- 非同期に変更される計算機資源の一貫性の保証
- 主にファイルの属性の操作に対する異機種透過性を実現すること
- ある条件で設定されているファイヤーウォールを越えられることコネクションを張る方向を問わない (passive mode と active mode)
- ユーザが所有権を持っている計算機資源はユーザの権限で export できること。
- ユーザ以外の誰か所有権を持つファイルは export できないこと。
- export しているファイルシステムの上の、所有権を持たないファイルは、透過的にアクセスできず、名前も見えないこと。

5.2 並列性透過性

位置固定型計算機の集合で分散計算機利用環境を構築する分散システムの場合、並列性透過性はそれらの計算機の間で同期的に協調して動作する仕組みを考慮することが主である。分散計算機利用環境に移動型計算機が加わると、非同期的に動作する計算機が、あたかも同期的に動作しているかのように見せる必要がある。

5.3 FTFSのプロトタイプの設計

5.3.1 利用上の制約

FTFSのプロトタイプでは実現しない機能を以下に述べる。

- FTFSのプロトタイプ実装で共有する計算機資源は静的な情報である、ファイルだけに限定する。デバイスファイルやFIFOの名前付きパイプ等のプログラムの実行を伴う計算機資源の共有は行わない。

5.3.2 FTFSのインターフェースの定義

FTFSの内部に存在する主要なオブジェクトのインターフェースを抽象的なオブジェクト指向言語によって記述する。

```
interface FileSystem {
  Resource  openResource(String path_name);
  Resource  openResource(String path_name, Version version);
  File      createFile(String path_name);
  Directory createDirectory(String path_name);
}
```

ノード上のファイルシステムは、ディレクトリを指定し、ディレクトリ以下に含まれる全てのファイルを共有する。

ファイルシステムはリソースの名前を指定して、リソースの実体にアクセスするための代理オブジェクトを得る。

名前管理部分ではファイルシステムを結合するためのメソッドを持つ必要があるため、このインターフェースクラスを継承したクラスが定義される。

```
interface Resource {
  Attribute getAttribute();
  Attribute setAttribute(Attribute new_attribute);
  Location  getLocation();
  bool      setMainLocation(Location location_of_node);
  bool      duplicate(Location location_of_node);
  Version[] getHistory();
}
```

Resourceはファイルとディレクトリを表すクラスのスーパークラスである。

```
interface Attribute {
  User          getOwner();
  Group[]       getGroup();
  Permission[3] getPermission();
  Location      getCreateLocation();
  Time[]        getTimes();
```

```
}



interface Version {
  String getVersionIdentifier();
}

interface User {
  String  getName();
  Group[] getGroup();
}

interface Group {
  String  getName();
}

interface File extends Resource {
  int  read(byte[] buf, int read_size);
  int  write(byte[] buf, int write_size);
  int  close();
}

interface Directory extends Resource {
  Resource[] getList();
}

interface Permission {
  bool  isReadable();
  bool  isWritable();
  bool  isExecutable();

  void  setReadable(bool true_of_false);
  void  setWritable(bool true_of_false);
  void  setExecutable(bool true_of_false);
}
```

FTFS は三つの機能部分から構成されている。上記のクラスは三つの機能部分で共有されるオブジェクトである。実装する際にはそれぞれの機能部分でこれら

のクラスを継承するクラスを作成していくことを想定している。

三つの機能部分の間での通信プロトコルを規定する代わりに、CORBA や RMI などの分散オブジェクトを実現する技術を使って実装する方法が 考えられる。

第6章　結論と今後の課題

6.1　全体のまとめ

本論文ではまず、移動型計算機が計算機利用環境として使われる状況が増えて行くことに対して、移動型計算機を分散計算機利用環境の一部として機能させることが必要であることを述べた。

移動型計算機を効率的に利用するための環境が移動型計算機利用環境である。移動型計算機が持つ特性によって制限されている機能を、ネットワーク上の位置固定型計算機で構成される分散システムが支援することで、より多くの機能を利用することが可能になることを述べた。

移動型計算機を支援するための機能と、分散システムによって実現可能な機能をまとめ、それらを実現することによって得られる移動型計算機利用環境を考えた。

そして、そのような移動型計算機利用環境を実現するために必要な機能をまとめ、移動型計算機利用環境を実現する分散システムの仕様をまとめた。分散システムを利用する際の現実性を考えて、分散システムは既存のシステムに対する拡張によって実現されるべきである。

最後に、この仕様に基づいて、情報資源を共有する機能部分の設計を行った。この機能部分は分散システムの基盤である。既存のシステムに対する拡張として実現したので、この機能部分は既存のシステムからはファイルシステムとして扱われる。さらに、分散システムとしての既存のファイルシステムとは異なる機能を利用するためには、ライブラリを提供し専用のプログラムを作ることで解決することを述べた。

6.2　FTFSのまとめ

FTFSは既存のシステムに対する変更を行わずに、分散システムによって実現される機能を利用することが可能である。これは既存のシステムに備わっている機能拡張のインターフェースにあわせた、追加システムによって分散システムの機能を実現しているからである。

FTFS では分散システムの機能として、位置透過性、障害透過性が実現されている。

名前管理機能をユーザの責任で実行することによって、管理ドメインを越えて

分散する情報資源に対して、位置透過性を実現する。
障害透過性は資源のバージョン管理によって実現している。FTFS では情報資源に対する変更を履歴として別々に保存して、バージョン管理を行っている。これよって、次の三つの機能を実現している。

- 移動型計算機がネットワーク切断状態でも情報資源への透過的なアクセスを行う機能。
- 移動型計算機がネットワーク切断状態でも非同期に情報資源を操作すること。その操作に対して情報資源の一貫性を保証する機能。
- 情報資源に対する信頼性を向上する機能。

これらの機能によって、情報資源を移動型計算機を含めた分散計算機利用環境で透過的に扱うことが可能になっている。
以下に長所と短所をまとめる。

6.2.1 FTFS の長所

- 管理ドメインを越えて分散する計算機の情報資源を透過的に扱う方法を提供していること
- ネットワーク切断状態でのアクセスの透過性を実現していること
- 情報資源の一貫性を保証していること
- 障害に対する信頼性を向上していること

6.2.2 FTFS の短所

他のユーザと情報資源を共有する際に、アクセスの透過性などの分散システムとしての長所を実現できていない。
資源利用部をシステムに組み込むためには、そのシステムの管理権限を必要とする。したがって、管理権限を持たない計算機上の機能資源が FTFS 上の情報資源へアクセスする方法が制限される。

6.3 今後の課題

6.3.1 移動型計算機利用環境を実現する分散システムに欠けている機能の実現

本論文で仕様を決定した分散システムは、ユーザとの計算機資源の共有に対して、透過性を実現できていない。ユーザごとに存在している仮想計算機間での計算機資源の共有を考える必要がある。

これは例えば、名前空間を管理する機能部分が、他の仮想計算機を構成する名前空間管理機能部分と協調して、グローバルな透過的な名前空間をつくる方法が考えられる。この時、残り二つの機能部分がそのようなより広い名前空間にアクセスし、計算機資源の供給を行ったり、受けたりできるような仕組みが考えられる。

6.3.2 FTFS だけでは実現できない問題

FTFS は移動型計算機を支援するための分散システムの、情報資源に対する機能部分しか透過的に扱うことができていない。移動型計算機を支援するための分散システムを完成させるためには、機能資源を透過的に扱う機能を実現しなければならない。

謝辞

本研究を進めるにあたり、御指導をいただきました、慶應義塾大学環境情報学部教授の徳田英幸博士と村井純博士に感謝いたします。また、絶えず御助言と御指導をいただきました、同学部の中村修博士と楠本博之博士に深い感謝の意を表します。

重近範之氏、土本康生氏をはじめとする慶應義塾大学政策メディア研究科後期博士課程の方々、並びに同大学環境情報学部の徳田・村井・中村・楠本研究室の諸氏には、本研究を進めていく上で様々な議論をして頂き、また耐えざる励ましとご援助を頂きました。ここに、深い感謝の念を表します。

以上を持って謝辞といたします。

2000 年 1 月
宅間 啓

参考文献

[1] Andrew Birrell, Greg Nelson, Susan Owicki and Edward Wobber: “Network Objects” Digital Systems Research Center

[2] 前川守・所真理雄・清水健多郎偏: 「分散オペレーティングシステム」共立出版株式会社

[3] Mohammad Amin Vahdat: “Operating System Services for Wide-Area Applications” GRADUATE DIVISION of the UNIVERSITY of CALIFORNIA, BARKELEY

[4] Amin M. Vahdat, Paul C. Eastham and Thomas E. Anderson: “WebFS : A Global Cache Coherent File System” Computer Science Division of the University of California, Berkeley

[5] 土門哲也、高清水直美、佐々木節: 「DCE による分散ファイルアクセスの研究」文部省高エネルギー加速器研究機構計算科学センター、日立情報システムズ

[6] 盾岡孝道、植原啓介、砂原秀樹、寺岡文男: 「PFS:通信環境に動的に適応するファイルシステム」コンピュータソフトウェア,Vol.15,No.2(1998),pp62-81, JSSST

[7] J. Veizades, E. Guttman, C. Perkins, S. Kaplan: “Service Location Protocol” Request for Comments 2165, June 1997

[8] Sun Microsystems, INC.: “Jini(tm) Connection Technology” http://www.sun.com/jini/

[9] “Home Audio/Video Interoperability” http://www.havi.org/

[10] A.S. タネンバウム: “OS の基礎と応用” 株式会社プレンティスホール出版, 1995

[11] Bell 研究所: “Plan 9 Index” http://plan9.bell-labs.com/plan9/

[12] Lucent Technologies Inc. “Inferno” http://inferno.lucent.com/inferno/

[13] Sony CSL, Inc. "Apertos: the Reflective Object-Oriented Operating System" http://www.csl.sony.co.jp/project/Apertos/

[14] R.E.Schantz, R.H.Thomas, G.Bono: "The Architecture of the Cronus Distributed Operating System" Proc. 6th Int. Conf. on Distributed Computing Systems, pp. 250-259, Cambridge, MA, May 1986

[15] B.Callaghan, B.Pawlowski, P.Staubach: "NFS Version 3 Protocol Specification" Request for Comments 1813, June 1995

[16] Carnegie Mellon "Andrew II Home Page" http://andrew2.andrew.cmu.edu/ANDREWII/AndrewII.html

[17] James H. Morris, Mahadev Satyanarayanan, Michael H. Conner, John H. Howard, David S.H. Rosenthal, F.Donelson Smith: "Andrew: a Distributed Personal Computing Environment" Communications of the ACM, Vol.29, No 3, March 1986

[18] Anthony D. Joseph, et al.: "Rover: A Toolkit for Mobile Information Access" Proc. of the Fifteenth Symposium on Operating System Principles, December 1995

[19] "RAM Mobile Data System Overview" RAM Mobile Data Limited Partnership USA RMDUS 031-RMDSO-RM Release 5.2, October 1994.

[20] J.J. Kister, M. Satyanarayanan: "Disconnected Operation in the Coda File System" Proc. 13th Symposium on Operating Systems Principles, 1991

[21] J.J. Kister, M. Satyanarayanan et al.: "Experience "

[22] D.M. Huizinga, K. Heflinger: "Experience with Connected and Disconnected Operation of Portable Notebook Computers in Distributed Systems" Proc. of the IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994