平成19年度 修士論文

ノードの稼働パターンを考慮したP2Pバックアップシステムの設計と実装

電気通信大学 大学院情報システム学研究科

情報システム設計学専攻

0650014 黒澤 研吾

指導教員 多田 好克 教授
渡辺 俊典 教授
植野 真臣 准教授

再提出日 平成20年2月19日

目 次

図 目 次

表 目 次

第 1 章

はじめに

データのバックアップはコンピュータを使う上で古くから行われている重要な作業であり、現在もその重要性は変わらない。磁気テープや光ディスクなどのバックアップ用の記憶メディアを用意して、定期的に作業を行うのが一般的である。しかし、バックアップにかかる手間とコストは大規模な組織ならまだしも、個人ユーザにとっては大きな負担と言える。個人ユーザにとってもバックアップは重要であるが、そのためにかかる手間・コストを嫌い、定期的・継続的なバックアップを行っていないのが現状である。また、記憶メディアにバックアップを取った後もこれらのメディアを管理するコストが発生するという問題もある。

その一方で、個人のコンピュータに目を向けるとハードディスクは必ずしも限界まで使い切られているわけではなく、比較的多くの空き領域が各所に遍在している現状がある。これらのコンピュータをネットワークで接続し、空きハードディスク領域を共有化することができれば共有バックアップシステムとして活用することができると考えられる。また、複数のハードディスクを使用しデータを分散化することにより、データの冗長性を保つことができ、バックアップメディアとして適切に扱えると考えられる。

しかしながら、個人のコンピュータは常時稼働しているわけではなく、稼働状態が定かでない。もし、共有バックアップシステムを構築したとしても、バックアップしたファイルがいつリストアできるかどうかはバックアップファイルが保存されているコンピュータの稼働状況に左右される。個人の生活リズムから得られる各

コンピュータの稼働状況を考慮することにより、短時間でのリストアやより少ないバックアップファイルの転送処理ができると考えられる。

人間の生活リズムは、「日」、「週」などの単位で周期的に繰り返されていることが多い。そこで、個人の利用しているパソコンは人間の生活リズムに相関があると考え、将来のコンピュータの稼働状況を類推することにした。本研究では、各コンピュータをネットワークで接続し、各コンピュータの稼働状況予想を使った P2P バックアップシステムの設計と実装を行う。

そこで、まず、各コンピュータの稼動状況を自動的に取得する。次にその稼働状況から将来稼働するであろう予想時間帯を推定する。推定した稼働予想を他のコンピュータと共有することによりどの時間帯にどのコンピュータに配置すればより短時間でのリストアが可能となるか、ファイルの配置・転送方法について考える。

本論文は、以下の章構成で書かれている。第 2 章に個人用としてよく使われているバックアップ方法について述べ、第 3 章に今回作成したシステムの概要を述べる。第 4 章にシステムの設計について、第 5 章にシステムの実装について議論する。第 6 章に実験と評価を行い、第 7 章に関連研究を述べ本システムとの比較を行う。第 8 章に問題点と課題を述べ、第 9 章に本論文をまとめる。

第 2 章

バックアップメディア

現在、個人用に使われているバックアップ方法の代表例を挙げ、その利点、欠点について述べる。

- 光メディア

  光メディアの代表的なものとして、CD-R や DVD-R が挙げられる。光メディアは、記録ドライブからの強いレーザ光の照射による熱によって記録層にデータを記録している。記録容量は、CD-R では 700MB 程度、DVD-R では 1 層のもので 4.7GB、2 層のもので 8.5GB である。書き換え可能な CD-RW、DVD-RW もあり、書き換え可能回数はおおよそ 1000 回とされている。

  以下に利点と欠点を述べる。

  – 利点

    * 安価である

      安いもので 1 枚 100 円以下で購入できる。

    * メディアがコンパクトである

    * ランダムアクセスが可能である

      バックアップデータの中から一部のファイルを取り出すことが可能である。

  – 欠点

* 容量に制限がある

  CD-R に記録できる容量は 650MB、700MB、DVD-R で記録できる容量は 1 層のもので 4.7GB、2 層のもので 8.5GB と決められており、それ以上の容量は記録できない。そのため、記録できる容量を超えるデータをバックアップする場合は、メディアが複数になることもある。また、1 ファイルが記録できる容量を超えるデータサイズの場合は、ユーザがデータを分割してメディアに記録しなければならない。

* メディアの管理コストがかかる

  光ディスクは記録面を保護する必要がある。記録面に傷やほこりがあると正しく読み取れなくなる恐れがある。また、太陽光が直接あたるところに長時間放置しているとデータが読めなくなることもある。そのため、傷やほこりを防ぐためにカバーをつける必要がある。また、高温多湿でない場所や直接太陽光のあたらない場所に保管する必要がある。

* 書き込み/読み込みに専用のドライブが必要である

  書き込み DVD には、DVD-R、DVD+R、DVD-RAM と規格が分かれている。そのため、それぞれの記録ディスク専用のドライブが必要となり、個々のユーザはどの規格に対応したドライブを所持しているかを自分で調べる必要がある。

* データを書き込む際に専用のソフト (ライティングソフトまたはレコーディングソフトと呼ばれている) を別途用意する必要がある

  光メディアは、データを書き込むための専用ソフトが必要となる。Windows XP[1]、Windows Vista[2] では標準で記録ディスクの書き込みを行うことができるが、これ以外の OS では専用ソフトを用意する必要がある。

* 読み取り/書き込み速度に限界がある

光ディスクはモーターにより回転させられ、レーザーによってディスクの読み取り/書き込みが行われているため、データアクセスに時間がかかる。また、ディスク中央と端とでは読み取り/書き込み速度が異なる。

- フラッシュメモリ

電源を切ってもデータが消えない不揮発性半導体メモリが使われている。フラッシュメモリを使った代表的な記憶メディアとしてUSBメモリ、SDメモリーカード、メモリースティック、xDピクチャーカード、コンパクトフラッシュが挙げられる。USBメモリはUSBポートに挿すことで使用することができるが、それ以外の上記にあげたメディアでは読み書きするためのリーダー(カードリーダーとも呼ばれている)にメディアを挿入することで使われる。数百MB〜数GBの容量があり、2008年1月現在で一番大容量なもので16GBのものもある。消去・書き込み可能回数が少ない物は数百回程度でその限界を迎える。その後、コントローラチップによる制御で消去・書き込みが特定ブロックに集中しないように改良されたが、それでも数万回から数百万回が限度である。

以下に利点と欠点を述べる。

- 利点
  * メディアがコンパクトである
  * ランダムアクセスが可能で比較的高速に読み書きできる
- 欠点
  * 容量に制限がある

保存できる容量は数百MB〜数GBであるため、それ以上の容量は保存できない。そのため、記録できる容量を超えるデータをバッ

クアップする場合は、メディアが複数になることもある。また、1ファイルが記録できる容量を超えるデータサイズの場合は、ユーザがデータを分割してメディアに記録しなければならない。

* メディアの管理コストがかかる

  接続する端子部が剥き出しになっていることが多いため、記録されているデータは静電気等によって破壊される恐れがある。そのため、端子部が外部と接触しないようにカバーなどをつける必要がある。

* 専用のリーダーが必要である

  USB メモリ以外のフラッシュメモリでは読み書きするためには、専用リーダーが必要となる。

- 磁気テープ

  粉末状の磁性体をバインダーと共にフィルム上に塗布した帯状磁気記録媒体で、磁化の変化により情報の記録・再生を行う。個人より企業で使われるケースが多い。容量は、DAT 160 で 80GB と大容量である。テープ 1 本では容量が不足する場合、オートローダと呼ばれる装置でテープを自動的に交換することができる。そのため、制限容量を超えるデータのバックアップを行うのを自動化できる。書き換え可能回数は、100 回程度である。

  以下に利点と欠点を述べる。

  – 利点

    * データ容量が大きく、メディアの容量当たりの単価が安い

  – 欠点

    * 速度が遅く、バックアップ・リストアにかかる時間が長い

      ファイルの容量によっては、半日または 1 日かかる。

* ランダムアクセスが不可能である

  シーケンシャルアクセスにしか対応していないため、テープに記録されている途中のデータを読み取りたい場合に時間がかかる。

* 繰り返しの書き換えに弱い

* ドライブの管理コストがかかる

  ドライブ本体は、定期的な清掃が必要である。

* メディアの管理コストがかかる

  テープの性質上、環境によっては劣化が起こる可能性がある。

- ハードディスク

  ハードディスクを専用のケースに入れ、外部端子よりコンピュータと接続することでデータの読み書きを行う。接続に、USB、IEEE1394、Ethernet を使用できるものもあるため、高速でデータの読み取り/書き出しを行うことができる。容量は、数百 GB~1TB と大容量である。ハードディスクを複数台用意することにより、RAID(Redundant Arrays of Inexpensive Disks) を使うことで冗長性を持たせ、耐障害性を高めることが可能である。

  以下に利点と欠点を述べる。

  – 利点

    * 大容量である

    * ランダムアクセスが可能で比較的高速に読み取り/書き出しができる

    * 管理コストがかからない

      湿気や熱などによる劣化がないため、管理コストを低く抑えることが可能である。

* ディスクの耐障害性が上がる

  冗長性を持たせることで、耐障害性を高めることが可能になる。

– 欠点

* 局所的な障害に弱い

  コンピュータ本体とバックアップのためのハードディスクが同一のエリア (地理的に同じ範囲) に置かれるため、災害によってハードディスクが損傷した場合、データの回復が困難になる。

- ファイルサーバ

本研究では、ファイルサーバはLAN(Local Aria Network) 内など限られた範囲内でファイルを共有するために設置されるサーバのことを指すと考える。コンピュータ (または、それと同等の機能を持つデバイス) に複数台のハードディスクを用意し、ネットワークを通じてデータの読み取り/書き出しを行う。容量も、パソコンに内蔵されているものと同等もしくはそれ以上のものを使うことができる。

以下に利点と欠点を述べる。

– 利点

* ネットワークを通じてデータの読み取り/書き出しを行うことができる

– 欠点

* サーバのコストがかかる

  ファイルサーバを構築するための知識が必要となる。また、構築後の管理・運用コストが必要となる。

* 局所的な障害に弱い

- オンラインストレージ

  オンラインストレージ（ファイル・ホスティング・サービスとも呼ばれている）は、サービスを行っているサーバの空きハードディスク領域をインターネット経由で貸し出し、ユーザがそこにデータを保存できるようにするオンラインサービスである。主なオンラインストレージとして、ファイルバンク [3]、マイキャビ [4]、Yahoo!ブリーフケース [5] が挙げられる。自宅で進めた作業内容をこのオンラインストレージに預け、違う場所でダウンロードして作業を再開したり、簡易的なバックアップとして利用できる。また、借りている容量を他のユーザと共有化することもできるため、地理的に離れた他のユーザとの共同作業を行う場合も使うことが出来る。Web ブラウザや FTP クライアントから利用できるものが多い。オンラインストレージで利用できる容量は数 MB〜数百 MB 程度のサービスが多い。

  以下に利点と欠点を述べる。

  - 利点

    * どこからでもアクセスが可能である

      グローバルネットワークでサービスを行っているため、インターネットに接続可能であればいつでもバックアップ/リストア可能となる。

    * 局所的な障害に強い

      データが保存される場所が、互いに遠く離れていることが多いため、局所的な障害が発生した場合でもすぐにリストアすることができる。

    * サービスが充実している

      多くのサービスに検索、バックアップ日時情報、ソート機能が組み込まれているため、これらの機能を使うことですぐにバックアッ

プしたデータを見つけ出すことができる。

– 欠点

* 記録できる容量に制限がある

数 MB〜数百 MB であるため、その容量を超えるデータは記録できない。複数のアカウントを取得したとしても、すべての記録容量を一元的に扱うことが出来ないため、分割して記録することになる。

* 送信できる容量に制限がある

一度に送信できるファイルサイズが決められていることが多い。このため、大容量のファイルをバックアップする際には、ユーザが送信制限サイズに分割した上で送信しなければならない。

* セキュリティの問題がある

多くのサービスが、アカウントとパスワードだけで認証している。そのため、他人のアカウントとパスワードを取得することでその本人になりすますことができ、セキュリティ的に問題である。

第 3 章

システムの概要

図 3.1 に本システムの概要を示す。

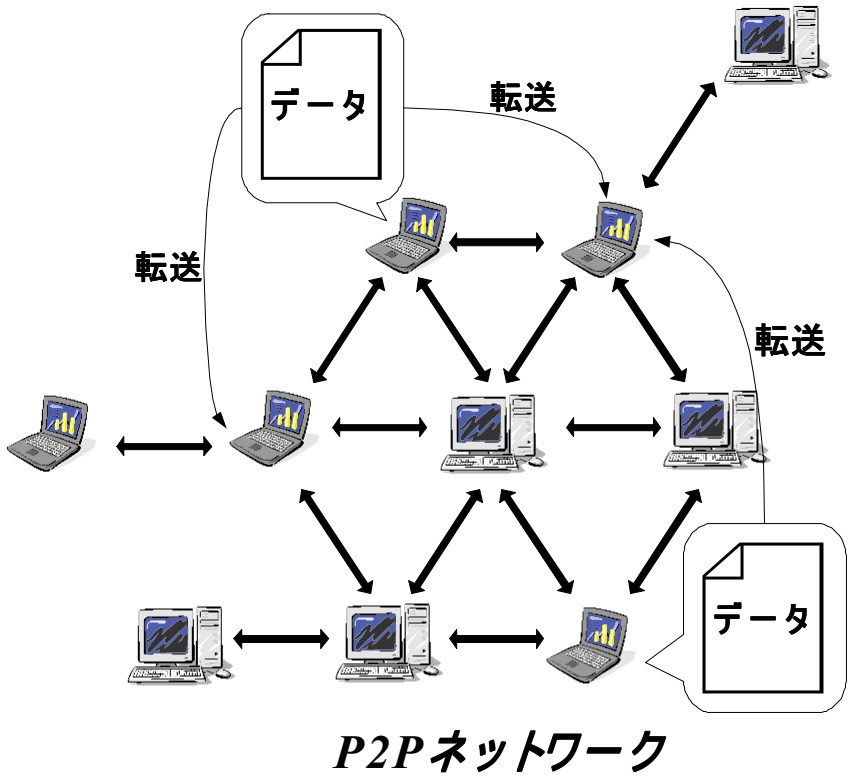

図 3.1: 本システムの概要図

本研究は、各コンピュータをネットワークで接続し、空きハードディスクを共有化し 1 つのストレージディスクとして見せる共有バックアップシステムを構築す

る。接続するネットワークモデルは、Pure Peer-to-Peer ネットワークを用いた。一般に Peer-to-Peer ネットワーク上でのコンピュータのことをノードと呼ぶ。以降、本論文もその表現を用いることにする。

- バックアップ処理

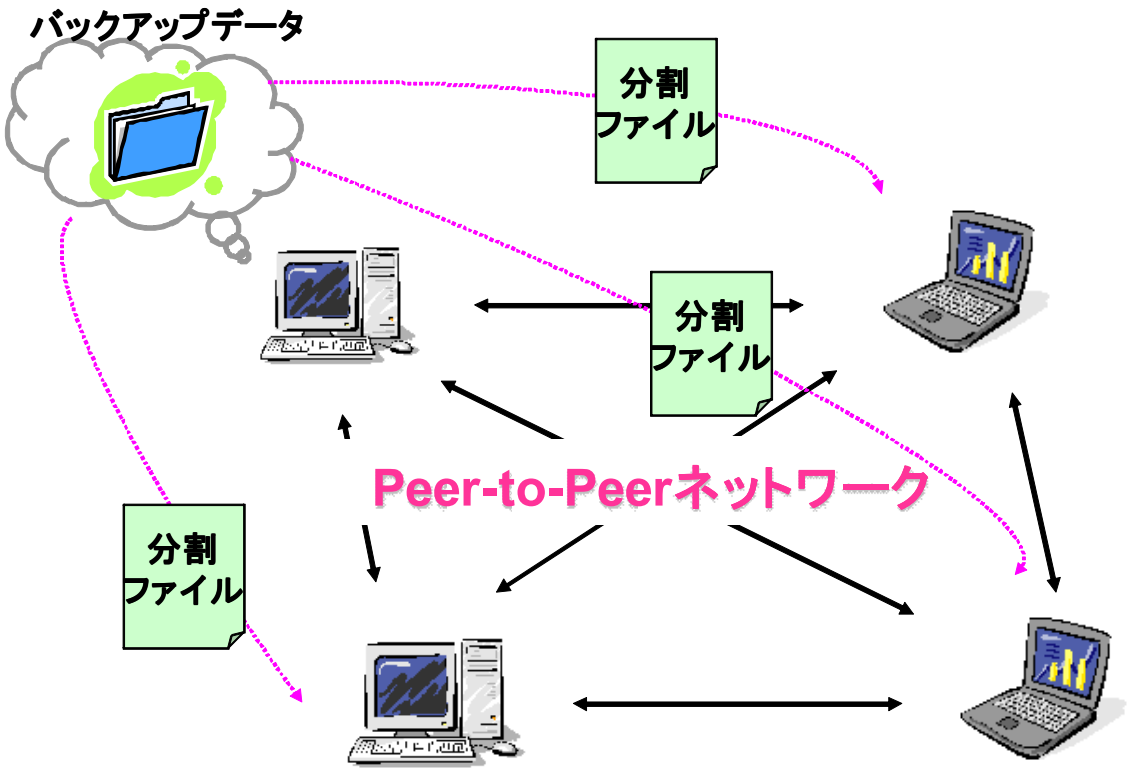

図 3.2: バックアップ処理

図 3.2 にバックアップを行う際の処理の流れを示す。バックアップを取ろうとするノードは、稼動している他のノードにバックアップファイルを分割し配置する。

- リストア処理

リストアしたいノードは、リストア要求を出し配置したノードよりバックアップファイルを取得する。リストアしたい場合に、バックアップファイルを所持しているノードが稼働しているかどうかで処理が異なる。そのため、ファ

イルを所有しているノードがリストア要求を出すノードと同時間帯に稼働する場合 (状況 a) とファイルを所有しているノードがリストア要求を出すノードと同時間帯に稼働しない場合 (状況 B) についての処理の流れを示す。

– 状況 a

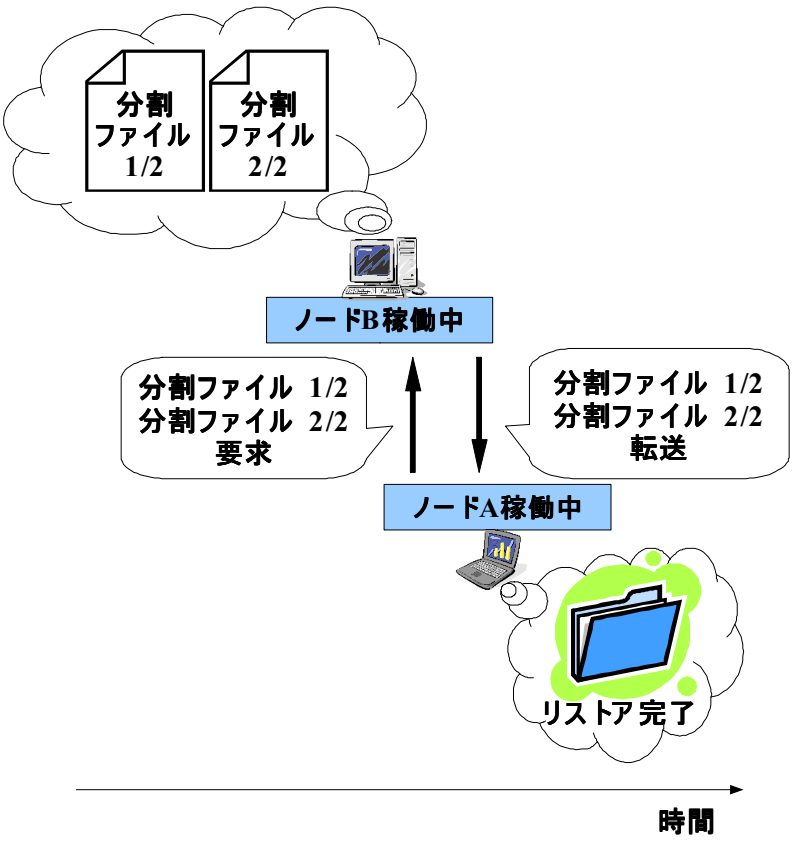

図 3.3: 状況 a の処理の流れ

状況 a の処理の流れを図 3.3 に示す。リストア要求を出したノード (図 3.3 中のノード A) と同時間帯稼動する他のノード (図 3.3 中のノード B) がファイル (図 3.3 中で分割ファイル 1/2、2/2) を所有している場合には、要求を出したノードにファイルが転送され復元される。

– 状況 b

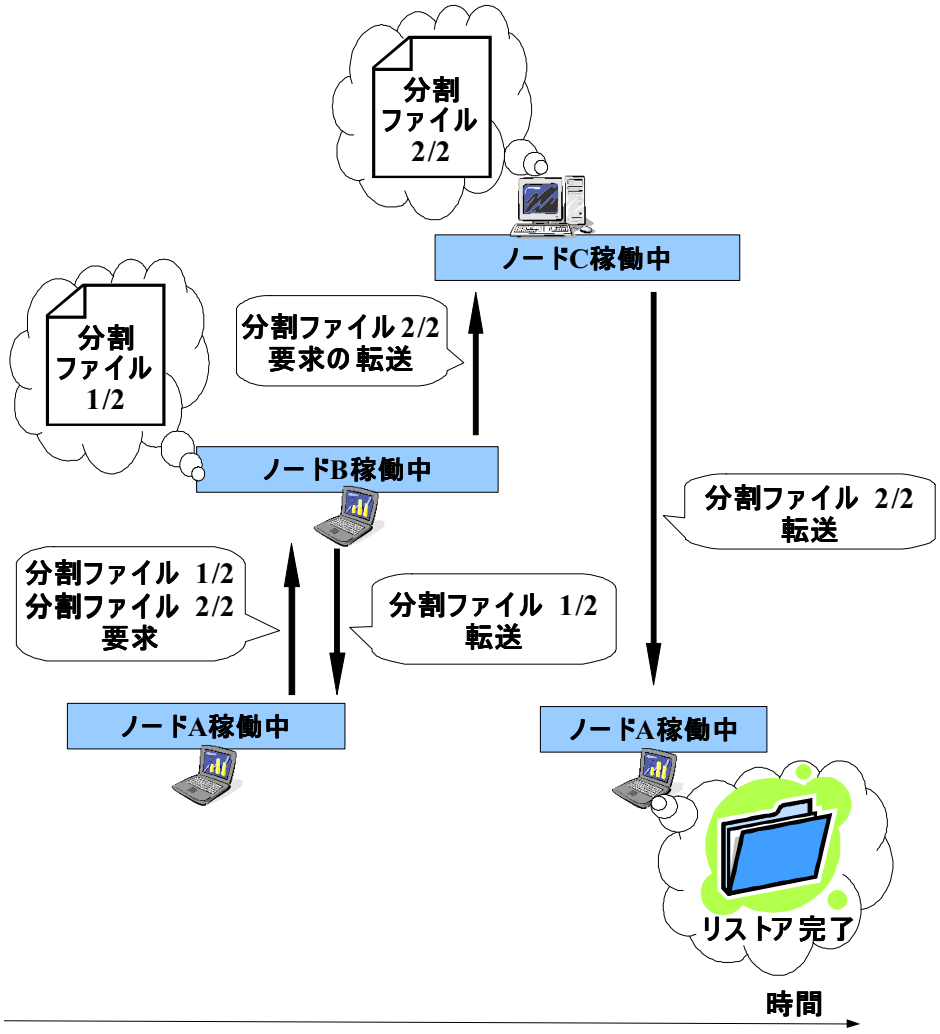

図 3.4: 状況 b の処理の流れ

次に、状況 b の処理の流れを図 3.4 に示す。リストア要求を出したノード (図 3.4 中のノード A) と同時間帯稼動する他のノード (図 3.4 中のノード B) がファイルを所有していない場合には、転送リストにファイル名が追加され、他のノード (図 3.4 中のノード B、ノード C) に転送される。その転送リストに記述されているファイルを所有するノードは、同時並行的に稼動するノードにバケツリレー方式でファイルが転送され、最終的にリストア要求を出したノードに転送され復元される。

第 4 章

システムの設計

この章では本システムを実現するための設計目標を示し、目的を達成するために本システムが解決しなければならない問題について議論する。その次に本システムの設計について記述する。

4.1 前提条件

本システムを使う上での前提条件について述べる。

- 想定する環境
  - 台数規模

    数台 ~ 数十台程度のコミュニティ内での使用を想定している。
  - ネットワーク環境

    WAN(Wide Area Network) を対象とする。また、本研究は、稼働状況からのリストアにかかる時間の短縮やファイル転送の回数を減らすことに重点を置いているため、接続しているネットワーク帯域はすべて同じとする。
- システム使用での条件
  - ノードが稼動している間は本システムが動作しているものとする

– あらゆる時間帯においてノードが孤立していない

– すべてのノードは固定 IP アドレスが割り振られており、すべてのノードがすべてノードの IP アドレスを知っている

– すべてのノードの時計は正確な時刻を刻んでいるものとする

– 同じユーザ名での本システムの使用は、ノード 1 つのみとする。

4.2 稼働状況

本システムでは、コンピュータの稼働状況を使って将来の稼動を予想する。そこで、コンピュータが「稼動している」/「稼動していない」を決める必要がある。

平日と休日における自宅のパソコンからのインターネット利用時間を図 4.1 に示す。

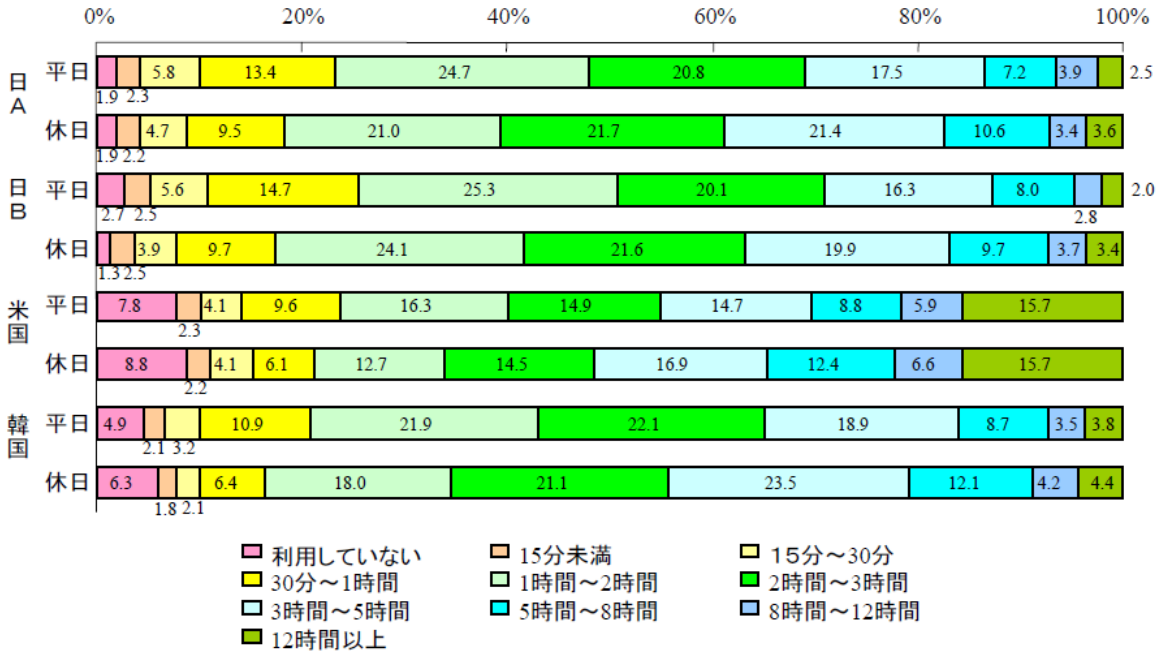

図 4.1: 自宅のパソコンからのインターネットの利用時間

(参考文献 [8] より引用)

自宅のパソコンからのインターネットの利用時間の調査は、ウェブアンケートを行い、日本国内で 2 種類 (日本 A、日本 B)、米国および韓国で 1 種類のデータを集計した [8]。これによれば、1 日の平均的なパソコンからのインターネットの利用時間は、日本 A が 184.0 分、日本 B が 175.3 分、米国が 293.5 分、韓国が 201.2 分と自宅で長時間パソコンを使っていることが分かる。また、会社でも勤務時間のほとんどをパソコンでの作業にあてていることが多いと思われる。パソコンの稼働時間を測り、その時間を勤務時間としている企業もある。多くの人が、長時間コンピュータを使っていることを考慮して、本システムでは 1 時間単位で「稼動」/「非稼動」とした。

4.3 ネットワークモデルの設計

本システムは、個人コンピュータをネットワークで接続し、ハードディスクを共有化する。そこで、本システムを構築するにあたり、適切なネットワークモデルを決める必要がある。ネットワークモデルを決定するための条件を以下に示す。

- 個人のコンピュータを対象とし、必ずしも常時稼働するノードが存在するわけではない
- 前提条件より接続するネットワーク帯域はすべて同じである
- 個人の作業の妨げにならないように、全ノードの負荷を均一化する必要がある

これらの条件より以下のことが言える。

- ユーザ名やファイルリストなどの情報を常時管理するノードが存在しない
- 一部のノードにファイル検索やファイル転送処理が集中することがない

これらのことより、本システムで用いるネットワークモデルとして、Pure Peer-to-Peer ネットワークモデルを用いるのが妥当であると考えられる。

4.3.1 Pure Peer-to-Peer ネットワーク

Pure Peer-to-Peer ネットワークは、中央サーバを必要とせず、ノードのみでネットワークを形成する。そのため、ファイルの検索や転送は直接ノード同士で行われる。各ノードは、隣接したノードに接続を行い、検索やファイルリストなどの情報をバケツリレー方式で順次ノードに転送する。利点としては、どこか一ヶ所が寸断されてもサービス全体が停止することはないため耐障害性に優れていることが挙げられる。一方欠点は、実装が複雑で、ノード数が増えると加速度的にネットワークが混雑し、帯域消費が増大することが挙げられる。

一般に Peer-to-Peer のことを P2P と省略される。以降、本論文もその表現を用いることにする。

4.4 設計方針

本研究で構築するシステムは、ノードの空きハードディスクを利用するバックアップシステムである。そのため、各ノードが多数のファイルをバックアップを行うようにするためには、ハードディスクにかかるコストをなるべく少なくする必要がある。そこで、本節ではバックアップファイルの複製の数、バックアップファイルの削除方法、稼働状況のプロファイリング、ファイル転送について議論する。

4.4.1 複製数

バックアップファイルの複製をすべてのノードに配置すれば、同時並行的に稼動するノードまたは自身から瞬時に復元することが可能である。しかし、各ノードのハードディスク容量は有限であることから、あまり現実的でないと考えられる。

本システムでの想定する環境下では、全ノードが稼動しているわけでなく、同時並行的に稼動しているノードが変動し、その数は増減する。そのため、バックアップファイルの分割を行い、同時並行的に稼動するノードに配置するようにした。

4.4.2 削除

本システムで保存できる共有ストレージ容量には限界がある。バックアップファイルの削除機能を組み込むことで、共有ストレージのディスクスペースを確保する。Free Haven[13]、Chord[14]、Freenet[15] のような P2P システムは、アクセスの少ないファイルを削除するか、ファイル失効日を設けてファイルを削除する。しかしながら、バックアップファイルはいつ必要になり、いつアクセスされるか分からないため、アクセスの少ないファイルを削除する方法とファイル失効日を設けてファイルを削除する方法は妥当ではない。そこで、バックアップを行ったユーザがどのファイルを削除したいかを決める方法とした。

4.4.3 稼働状況のプロファイリング

人間は、社会的生活を営む上で、ある生活リズムで行動することが多いと言われている。そこで、人間の生活リズムに着目し、過去の稼働状況を元に将来の稼働予想を求めようにした。

生活リズムについて

人間は、社会的生活を営む上で、ある生活リズムで行動することが多い。また、複数の種類の時間周期 (短い時間の周期と長い時間の周期) を同時に経験しながら日常生活を行っている [7]。

図 4.2 に示すように、人間の生活周期は 1 日、1 週間、1ヶ月、四季、1 年といった時間周期を繰り返している。そのため、過去の時間周期性を参照することで、将来の生活リズムを予想できるのではないかと考えられる。このような周期的な生活に着目し、過去の稼働状況を自動的に取得し、それらの稼働状況から将来の「稼動」/「非稼動」を予想する。

本システムで生活リズムの周期性を扱う上で、どの時間周期単位を使うのかが重

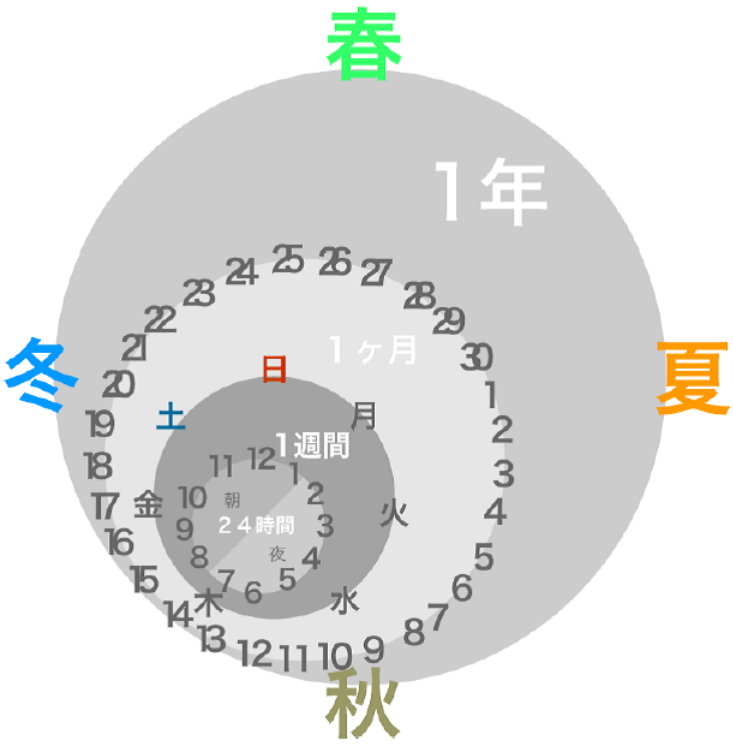

図 4.2: 人間の生活周期の概念図

(参考文献 [6] より引用)

要になる。これを決めるにあたり、万人が共通の周期・単位として扱っているものを定めるのが妥当だと考えられる。そのため、あいまい性のあるもの、ある地域にしか存在しない周期・単位や地形・空間に左右されるものは扱わないものとする。

本システムにおいて生活リズムを利用する上で、以下のことを決める必要がある。

- 本システムを使う上での最小時間単位 (最小稼働時間単位)
- 本システムを使う上での生活周期単位
- 本システムで使う将来の稼働状況の予想に使うデータの周期間隔

本システムを使う上での最小単位 (稼働状況の最小単位) について

第 4.2 節で述べたように、多くの人が長時間コンピュータを使っている [8] ことを考慮して、1 時間を本システムで扱う稼働状況の最小単位とする。

本システムを使う上での生活周期単位

今日の社会では、教育機関では週周期でカリキュラムが組まれていたり、就業やテレビ番組の週割り編成でも週を基本とした周期になっている [7]。そのため、幅広い人間が週を単位として生活していることがわかる。そこで、本システムでは 1 週間を生活の周期単位として決めた。

本システムで使う生活情報 (稼働状況) の予想に使うデータの周期間隔

どのような周期間隔で生活しているか現在の人によって異なる。また、同じ人でも曜日によって周期間隔が異なることがある。そのため、各々のノード (ユーザ) が自身の曜日毎の生活周期間隔を求める必要がある。そこで、ノードの過去のデータの自己相関を求めることでそのノードが稼動する周期間隔を求めることにした。

自己相関

自己相関は、データがそれ自身を時間シフトしたデータとでどれだけ良く整合するかを測る尺度であり、時間シフトの大きさの関数として表わされる。時系列データ分析で時間的相関（自己相関）が使われることが多く、自己相関を求めることで、時系列データに含まれる繰り返しのパターンを見つけることが可能である [9]。

1次の自己相関係数は、図4.3に示されるように1期ずらした相関係数である [10]。

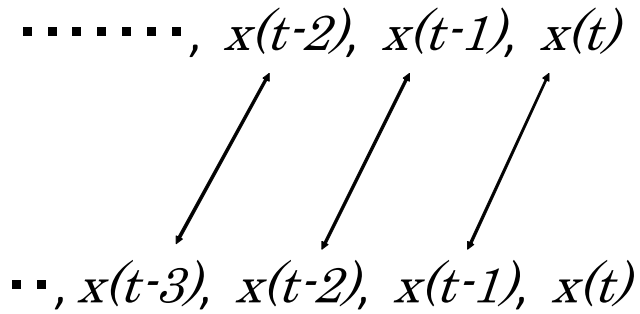

図 4.3: 1次自己相関

これを1、2、3、4、…、$n-2$ までの自己相関係数をとることでノードが稼動する周期間隔を求めることにした。

以下の式で自己相関係数 r を求めることが出来る。

$$r = \frac{\sum_{i=1}^{N-h}(x_i - \bar{x})(x_{i+h} - \bar{x})}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

$\bar{x}$: x の平均値（$1 \leq i \leq N$）

$$-1 \leq r \leq 1$$

5.3節では、自己相関を使った稼動予想法を説明する。

4.4.4 ファイルの転送

稼働状況によっては、ファイルを所有するノードと転送先の目的ノードが同時に稼動しない可能性が生じる。ファイルの転送する場合は、転送するファイル名がかかれたリスト (本研究では転送リストと呼ぶ) によってファイルの転送が行われる。図 4.4 にバケツリレー方式によるファイル転送を示す。ファイルを所有するノードから目的のノードまでのファイル転送は、バケツリレー方式で行う。

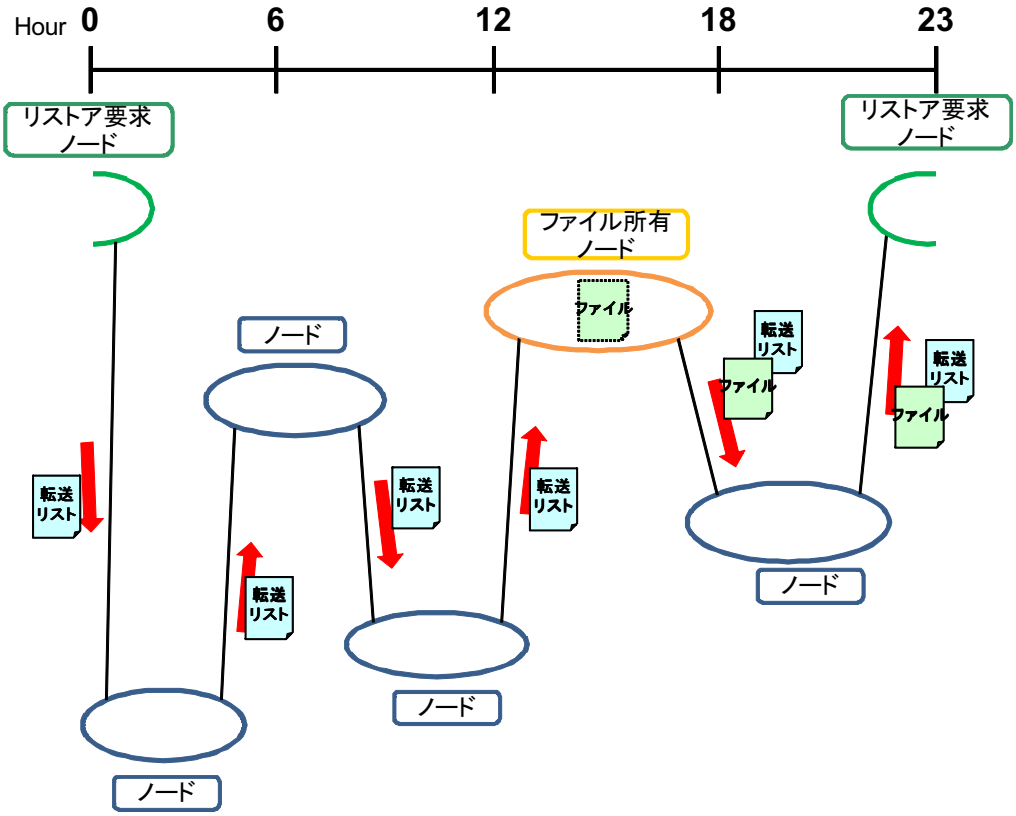

図 4.4: バケツリレー方式によるファイル転送

時間帯が異なって稼動する場合には、ファイルを所有するノードがその時間帯に同時稼動するノードにファイルを転送することにより目的のノードまでファイルが転送されることになる。

ファイル転送を行う際に、バックアップファイルの複製をすべてのノードに転送すれば、目的のノードに確実に転送することが可能となる。しかしながら、ファ

イル転送により各ノードの共有ディレクトリに一時的にファイルが置かれることになる。そのため、そのファイルの影響で容量不足になり、その時間帯のノードがバックアップファイルを配置できなくなる可能性が生じる。そこで、この問題を解決するために将来の稼働予想を用い、ファイル転送に使うノード数を減らすことにする。

第 5 章

システムの実装

5.1 実装概要

本研究は第 4 章での設計を基に、本システムの実装について述べていく。使用言語は C を用い、多くの UNIX 系 OS でも使用できるように実装した。また、本システムの実験を行う環境として、OS は Linux、FreeBSD を用いた。本システムの合計プログラム数は約 5300 行ある。P2P システムがベースプログラムであり、主なモジュールとして、ユニークなファイル名の形式、稼働状況のプロファイリング、ノードの配置・転送スケジューリングがある。各プログラム行数は、表 5.1 に表す。

表 5.1: プログラムの行数

システム	行数
ユニークなファイル名の形式	約 200 行
稼働状況のプロファイリング	約 260 行
ノードの配置・転送スケジューリング	約 770 行
P2P システム	約 3200 行

本章では、以下の機能の実装について述べる。

- ユニークなファイル名の形式

各ユーザがバックアップを行ったファイル名を P2P ネットワークに流れるファイル名に変換する。

- 稼働状況のプロファイリング

  過去の稼働状況より将来の稼働予想を求める。

- ファイルの配置・転送

  バックアップするファイルをどのノードに配置を行うかを決める。また、どのノードにファイルを転送するかを決める。

5.2 ユニークなファイル名の形式

5.2.1 バックアップファイル名

各ユーザがバックアップするファイルは様々であり、そのファイル名も様々なものがあると考えられる。そのため、ファイル名をそのままにしてネットワークに流すと異なるユーザが流した互いに異なるファイルのファイル名が衝突して、一方のファイルがもう一方のファイルに上書きされる可能性が生じる。そこで、ファイル名が衝突しないように、バックアップされるファイルのファイル名を変換することにした。この変換には、ユーザ名、バックアップ時刻、ハッシュ値、衝突回避値を用いた。図 5.1 にバックアップファイル名の形式を示す。また、その詳細を以下に述べる。

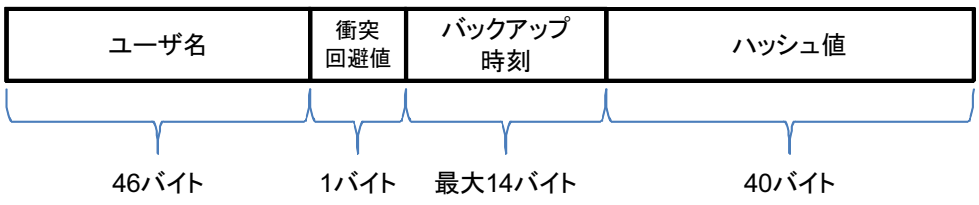

図 5.1: バックアップファイル名の形式

- ユーザ名

  ユーザ名はP2Pネットワーク上でユーザを一意に識別するため使用する。しかしながら、Pure P2Pネットワーク上で、ユーザ名を管理するノードがないため、ユーザ名を一元的に管理するのは困難である。一意に識別できる方法としてユーザ名を生成する方法の一つとして、MACアドレスなどのノードに依存する方法がある。しかしながら、ユーザとノードは1対1対応するとは限らない。そのため、ユーザ名は他のノードでも利用できるようにするため、ノードに依存する情報は避ける必要がある。そこで、UUID(Universally Unique Identifier)を使うことにより、本システムでのP2Pネットワーク上で使用するユーザ名を決定した。

  - UUID

    UUIDは、コンピュータに対するIDの統制を取れないような分散システム上で、一意に特定可能な識別子である。そのため、本システムで使うPure P2PネットワークでUUIDを使って一意のユーザ名を決めるのは妥当であると考えられる。UUIDは16バイトの数値で表され、理論上256^{16}(おおよそ3.4×10^{38})通り存在する。そのため、偶然、UUIDが一致することは起こりにくい。UNIXのコマンドラインのuuidgenを用いてユーザ名を生成した。出力結果は、16進数で表わされる。

- バックアップ時刻

  バックアップコマンドが入力された時刻をバックアップ時刻として使用する。バックアップ時刻として使用する時刻は、「年」、「月」、「日」、「時間」、「分」、「秒」である。

- ハッシュ値

  ファイル・コンテンツを元にSHA-1(Secure Hash Algorithm 1)を使って、

生成されたハッシュ値を使用する。UNIX のコマンドラインの sha1 を使ってハッシュ値を生成した。ファイル名に組み込まれたハッシュ値を利用することで、ファイル転送後のファイルの整合性をチェックする。出力結果は、16 進数で表わされる。

- 衝突回避値

  ユーザ名、バックアップ時刻、ハッシュ値を用いただけでは、ファイル名が衝突する恐れがある。これを回避するために、0~2 の値をとる衝突回避値をファイルの一部として組み込むことにした。この値は、デフォルト 0 であり、UUID、バックアップ時刻、ハッシュ値で変換する。変換されたファイル名が重なった場合にのみ衝突回避値をインクリメントする。本システムでは、バックアップコマンドが入力された時刻から次に入力できるまでに 1 秒以上かかるように設定している。また、4.1 節において、同じユーザ名での使用は、ノード 1 つのみとしている。このため、衝突回避値は 2 より大きい値を取る事はない。

5.2.2 最大ファイル名長の制限

ファイル名の長さはファイルシステムによって制限されている。この制限はファイルシステムごとに異なる。本実験を行う環境下 (EXT3、UFS) では、ファイル名長は最大 256 バイトに制限されている。5.2.1 節に述べた方法によるファイル名のユニーク化を行うと図 5.1 に示した通り最大で 101 バイトなので、ファイルシステムによるファイル名長の制限内に収まる。

5.2.3 メタデータ

UNIX のメタデータに関しては、UNIX の tar コマンドを使ってバックアップファイルを圧縮することで中に UNIX メタデータに組み込むことができる。そのため、

本システムで使うメタデータのみついて述べる。本システムは、バックアップを行った際にメタデータを出力する。このメタデータは、リストア時に必要な情報を含んだものであり、ユーザが行ったバックアップファイル名と 5.2.1 節で議論したファイル名を関連付けるものである。変換後のファイル名を知らないため、ユーザは変換前のファイル名を入力することでリストアする。また、メタデータはバックアップを行ったユーザのローカルディスクに保存される。そのため、メタデータは別途メディアに保存する必要がある。実際のメタデータの例を以下に示す。

```
[ken@gnu 0114]$ more BACKUP_FILE_LIST
file:test.dat
sha1 hash:bcd66d0823c36c7c86ba84564cb0c970dca2e1e8  test.dat
padding size:0
time:2007/12/21 17:10:44
split file:5719f87a-936e-11dc-9ce1-0010c60556c702007122117104405a3e0f1bfda819e52cf9f8120de2d0cc60e303f
split file:5719f87a-936e-11dc-9ce1-0010c60556c712007122117104405a3e0f1bfda819e52cf9f8120de2d0cc60e303f
split file:5719f87a-936e-11dc-9ce1-0010c60556c702007122117104405fe405753166f125559e7c9ac558654f107c7e9

[ken@gnu 0114]$
```

以下、このメタデータに含まれる情報の各要素について述べる。

- file : バックアップを行ったファイル名

  ユーザがバックアップを行ったファイル名

- sha1 hash : バックアップを行ったファイルのコンテンツ・ハッシュ

  ユーザがバックアップを行ったファイルのバックアップファイルのコンテンツ・ハッシュ

- padding size : 3DES-CBC で暗号化する際のゼロパディングサイズ

  3DES-CBC で暗号化を行う際に、8 バイトに合せるためにゼロパディングサイズ

- time : バックアップを行った時刻

  バックアップを起こった際の時刻

- split file：P2P ネットワークに流れるファイル名

  5.2.1 節で変換されたファイル名。ユーザ名、衝突回避値、バックアップ時刻、ハッシュ値で構成される。例を図 5.2 に示す。

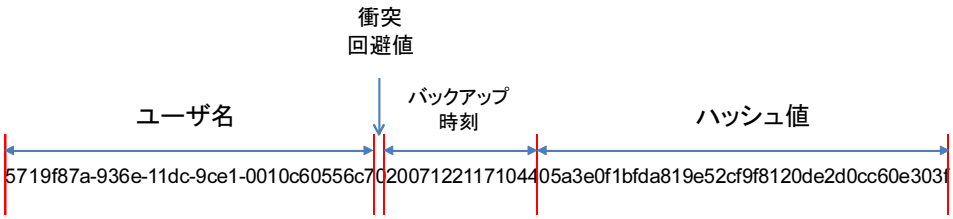

図 5.2: split file

5.3 稼働状況のプロファイリング

毎日各々のノードで稼働状況を取得する。4.2 節で述べたように、本システムでは、稼働状況の最小単位を 1 時間としている。ある 1 時間中の稼働が 30 分以上であれば、「稼動した」という情報が記録される。逆に、その 1 時間中の稼働が 30 分に満たない場合や全く稼動しない場合、「稼動しなかった」という情報が記録される。

記録された過去の稼働状況は将来の稼働予想を行うために用いられる。以下にそのアルゴリズムを示す。

step 1 本システムにより 1 日単位で稼働データを保存する (図 5.3)

step 2 保存された過去の稼働データを曜日毎にまとめ、一連の稼働状況データとする (図 5.4)

step 3 一連の稼働状況データを 1 週間単位で時間シフトを行い、1～N-2 次の自己相関係数を計算する。計算された自己相関係数の中で一番高い値よりそのノードの稼動周期間隔を割り出す (図 5.5)

step 4 *step* 3 で求められた稼働周期間隔上で、現在はその何週目に当たるかを見つけ出し、稼働周期より将来の稼働予想を決める (図 5.6)

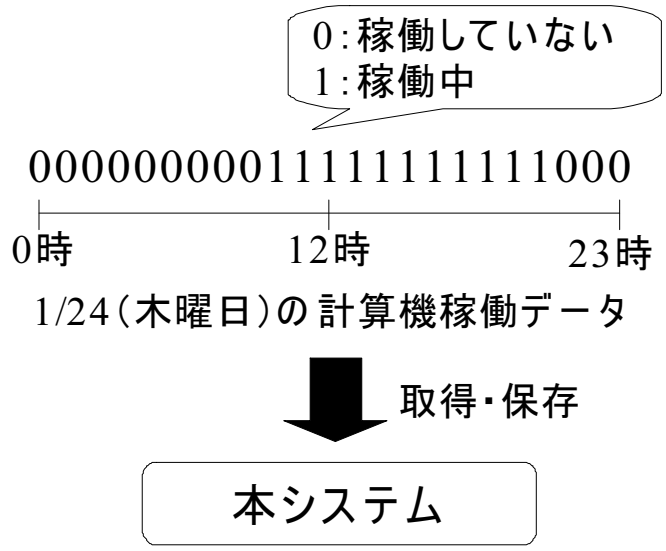

図 5.3: *step* 1 の処理

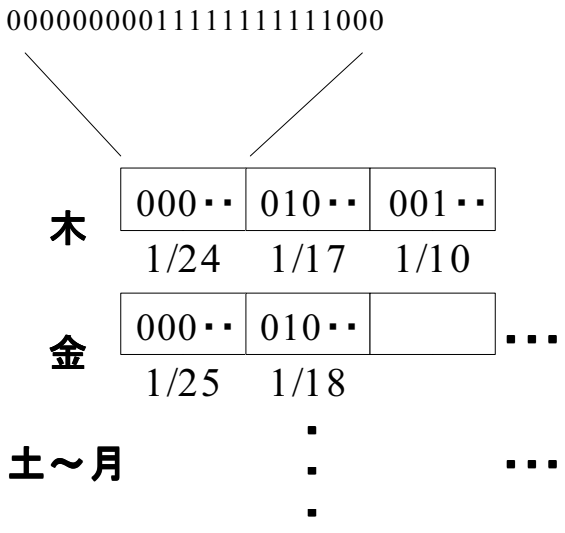

図 5.4: *step* 2 の処理

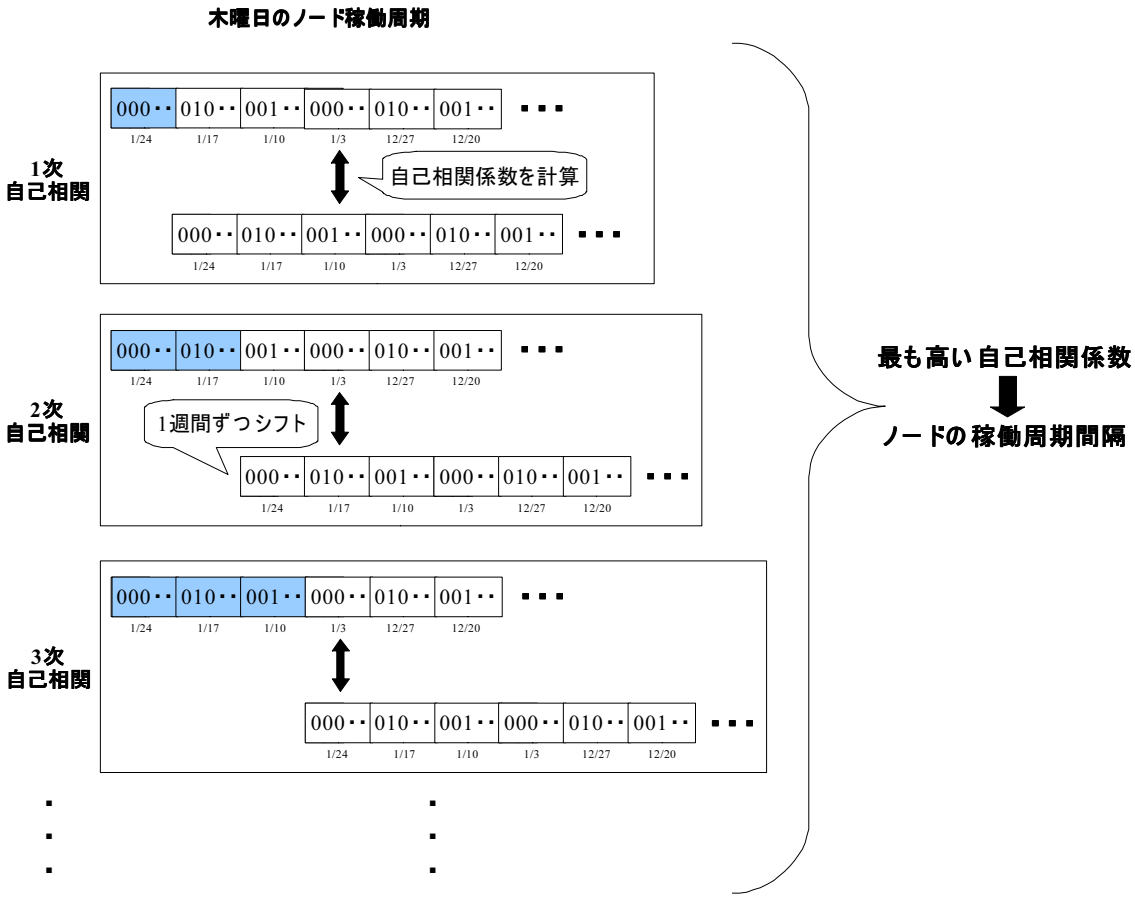

図 5.5: *step* 3 の処理

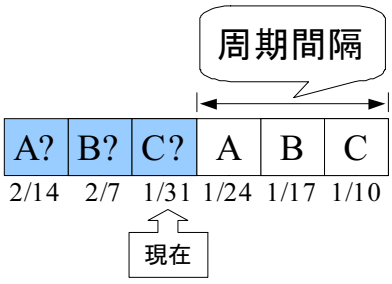

図 5.6: *step* 4 の処理

5.4 ファイルの配置・転送

5.4.1 ノードのグループ化

5.3 節で求めた稼働予想データが P2P ネットワークで共有されるように、本システムでは稼働予想を P2P ネットワークに流す。本システムは、バックアップやファイルの転送を行う際に同時稼動するノードを見つけ出し、ある程度の数のノードをグループ化する。5.4.2 節で述べるように、バックアップファイルを 2 分割し、分割ファイルよりパリティを計算する。それら 3 つのファイルをグループ化した自身以外の 3 つのノードに 1 つずつ配置するため、1 グループのノード数を 4 とした。グループ化を行ってバックアップ/リストア、ファイル転送を行うことにより、リストアにかかる時間が短縮でき、また、ファイルの転送コストを下げることが可能であるのではないかと考えた。

稼動予想と所属グループ数を用いて、着目するノードが属すべきグループを見つけることにした。着目するノードとは、バックアップを行う場合には自身ノード、ファイルを転送する場合には転送する際に稼働するノードを指す。以下にノードのグループ化アルゴリズムについて述べる。

step 1 着目するノードの稼動予想時間内に同じく稼動する他のノードを見つける

step 2 *step* 1 で見つかったノード群の中で稼動予想の重複時間が最も長いノード (群) を探す

case 1 見つかったノードが 3 つ以上の場合

step 3 見つかったノードにおいて、それぞれの所属グループ数 (そのノードが既に所属しているグループの数) を調べる

step 4 所属グループ数の少ないノードを優先的に選択する (所属グループ数が同じ場合は、先頭から選択される)

case 2 見つかったノードが 1 つ以上 3 つ未満の場合

step 3 見つかったノード (群) をグループの候補とする

step 4 *step* 2 で見つかったノード群の中で稼動予想の重複時間が次に一番長いノード (群) を探す

step 5 *step* 3 でグループの候補になっているノード数と *step* 4 で見つかったノード数を合わせた数が 3 つ以上の場合には *case* 1 に、3 つ未満の場合には *case* 2 を行う

case 3 ノードが見つからなかった場合

case 3.1 グループ候補となるノード数が 1 つ以上ある場合

step 3 グループ候補の中で着目しているノードとより稼動予想の重複時間が長いノードに着目ノードを変更して、*step* 1 を行う

case 3.2 どのノードとも稼動予想が重複していない場合

step 3 1 日後の曜日の稼動予想を用いて、*step* 1 を行う

5.4.2 ファイルの配置

5.4.1 節で述べたグループ化されたノード同士でバックアップを行う。分割されたバックアップファイルは、自身以外の各ノードに配置されるようにした。しかしながら、空きハードディスクの容量が少ないために、あるノードはこの分割ファイルを保存できない可能性がある。そこで、この場合には、そのノードを除いて一番長く同時稼動するノードに保存しきれなかった分割ファイルを配置することにした。

共有ストレージのサイズを減らすために、1 つのノードに分割したファイルを複数配置するのではなく、分割ファイルを自身以外の 3 つの各ノードに 1 つずつ配置するようにした。また、バックアップファイルのリストアできる確率を高めるために、分割された 2 つのファイルより計算されたパリティを計算する。そのパリ

ティ・ファイルと分割された2つのファイルをグループ化したノードの自身以外の3つの各ノードに1つずつ配置する。

5.4.3 ファイルの転送

4.4.4節で述べたように、稼働状況によっては、ファイルを所有するノードと転送先の目的ノードが同時に稼動しない可能性がある。そのため、他のノードにファイルを転送することにより、目的ノードへファイルを転送するようにした。

ノードからノードへのファイル転送ルートは、稼働予想データより計算される。このとき、転送命令を出したノードが全転送ルートの計算をするのではなく、終了する時間に転送するノードを探し出す。これにより、1つのノードが全部のルート探索を行う必要がなくなる。

転送するノードを決めるにあたり、その時間帯前後に一番長く稼働するノードに転送することにより転送コストおよびディスク容量コストを抑えることができると考えられる。稼動予想が確実であり、転送されたノードがその時間帯に問題なく稼動すれば、この方法が適切である。しかしながら、稼動予想に反して稼動しないケースや転送されたノードが故障してしまうケースが起こる可能性がある。そこで、このような場合でも対処できるようにするため、グループ化されたノード群によるファイル転送を行うことにした。その際、グループ化された各々のノード同士でファイルのコピーが行われる。どのグループに転送を行うかは、ファイルを所有しているグループの中のノードの1つが稼働予想を用いて計算する。また、ファイルを転送しようとするノード群にすでに転送するファイルが存在する場合には、転送処理は行われない。

ファイル転送の例を図5.7に示す。

図5.7に示したように、グループ化されたノード群にファイルを転送する。グループ化されたあるノード群に含まれる全ノードにファイルがコピーされる。また、転送されるノードにファイルが転送されると他のノードからの転送処理は行

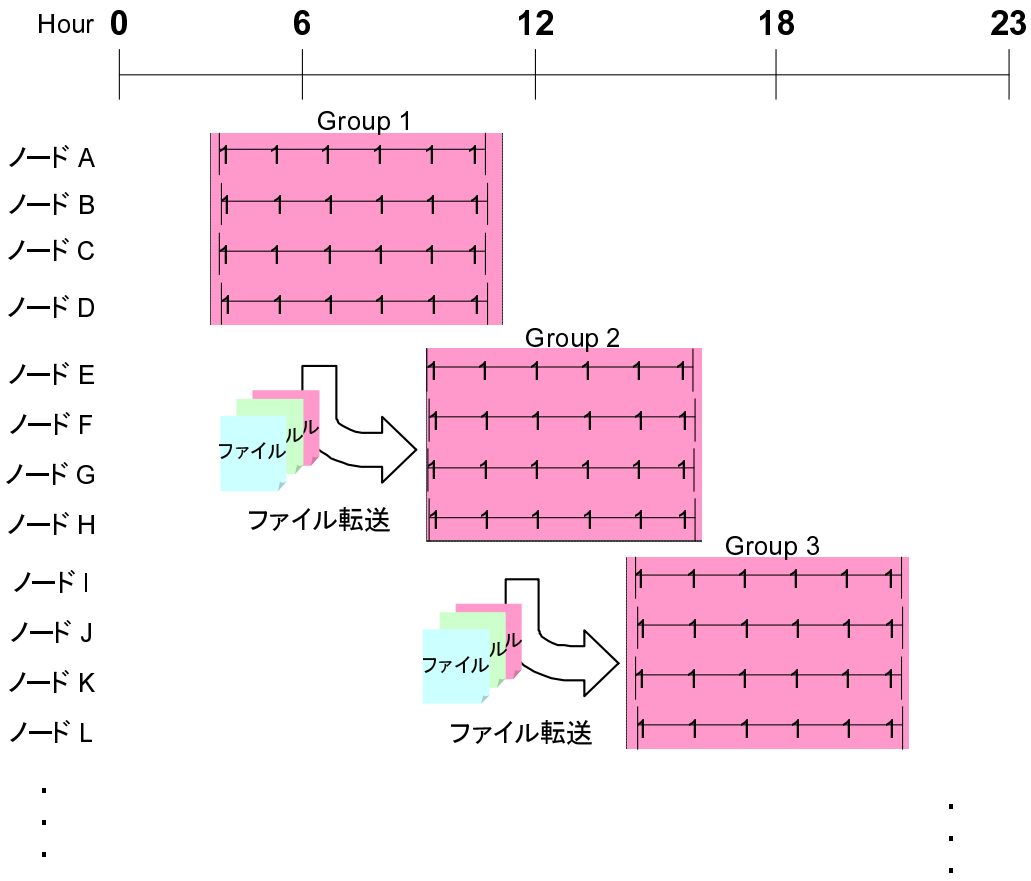

図 5.7: ファイル転送の例

われないため、Group 1 のどれかのノードが Group 2 のどれかのノードにファイルを転送すると、それ以外のノードはファイル転送を行わない。

同時間帯のすべてのノード群でファイルの転送を行うと、転送コストが増大する問題が発生する。そこで、転送コストを抑えるために、同時間帯でのうち 1 つのノード群にファイル転送を行うようにした。同時間帯に複数のノード群が存在する場合には、図 5.8 に示すように、その時間より後の時間帯で一番長く稼動するノード群に転送されるようにした。

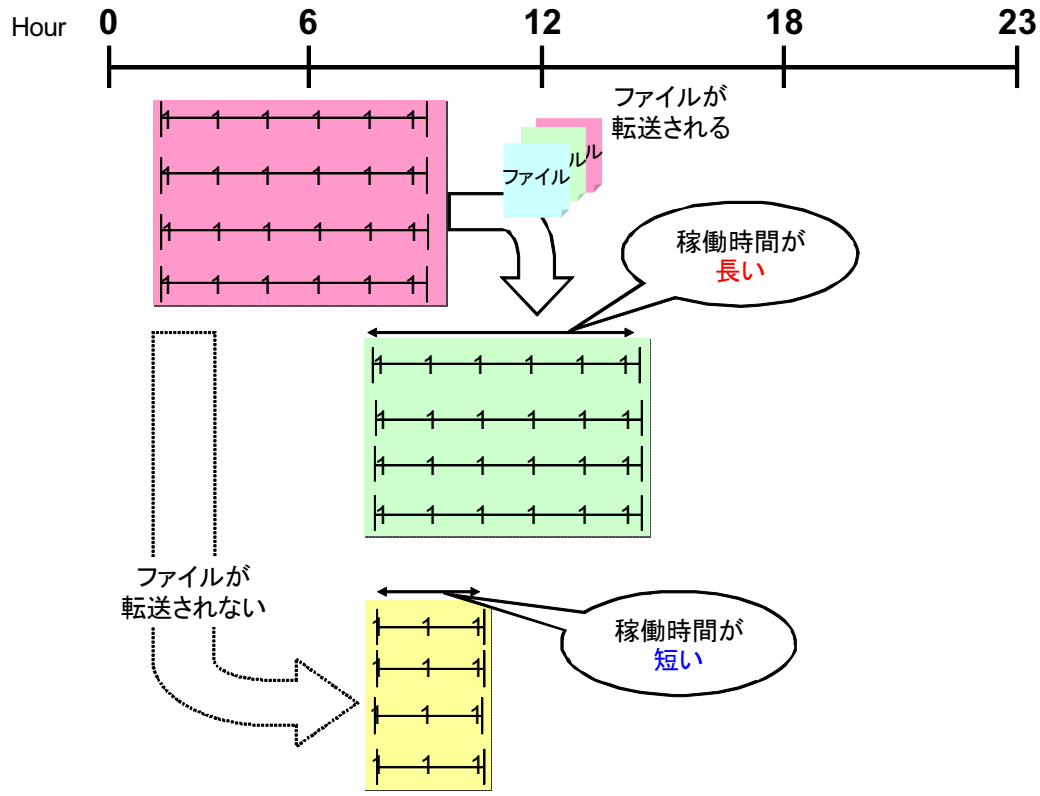

図 5.8: ファイル転送を行うノード群の決定

なお、ファイル転送は以下の場合に行われる。

リストア命令時に、接続するノードにバックアップファイルがない場合

リストア要求を出した時に、接続しているノードにファイルがない場合には、転送リストにリストアしたいファイル (分割されたファイル) の名前が追加され、他

のノードに転送リストが転送される。転送リスト内のファイルを所有するノードは、リストア要求を出したノードにファイルが転送されるように転送処理を行う。また、リストア要求後にファイル転送が行われることを本研究では、「後転送」と呼ぶことにする。

配置したいノードが稼働予想に反して稼働していない場合

バックアップを行う際に、稼働予想から求められたノードが稼働予想に反して稼働していない場合には、分割ファイルをノードに置くことができなくなる。そこで、その時間帯に稼働しているノードにファイルが配置され、稼動予想に反したためにファイルが配置できなかったノードへファイル転送を行う。こうすることで、本来、配置されるべきノードにファイルが転送される。

稼働予想に反して稼働した時にバックアップを行う場合

稼働予想に反して稼働した場合には、ファイルを配置するノードは稼動予想から見つけられないことになり、ファイルをノードに置くことができなくなる。そこで、バックアップを行ったノードと全時間帯で一番長く同時並行稼働するノード群を稼動予想より見つけ出す。見つけ出されたノード群にファイルを転送するため、その時間に稼動しているノードに一時的にファイルを配置し、ファイルの転送を行う。また、このファイル転送のことを本研究では、「前転送」と呼ぶことにする。

第 6 章

実験と考察

本システムの有用性を確認するために実験を行った。

6.1 稼働データ

10~20 代に尋ねた平均的なパソコンの利用時間帯情報を稼働データとした。なお、今回得た稼動データは個人生活を特定できる可能性があるため、本人の了解を得てデータを使用した。

6.2 実験

VMware 上で仮想マシンを用いて本システムのアルゴリズム有用性を確かめる実験を行った。実験環境、ノード条件、転送条件、実験内容については以下の通りである。

- 実験環境
  - 仮想マシンの台数：9 台
  - 仮想マシンで使う OS：fedora 8(Linux 2.6.23.1-42.fc8)、CentOS 4(Linux 2.6.9-55.0.2.EL)、FreeBSD 6.1(FreeBSD 6.1)
  - 使用したデータ期間：13 週間

– バックアップファイルサイズ：107,696,436 バイト

- ノード条件

＜1＞ 稼働予想を使い、グループ化されたノード群に複製を配置する

転送機能は、本システムで組み込んだ稼動予想によるノード群のファイル転送アルゴリズムでファイル転送を行った。

＜2＞ ランダムに複製を配置する

転送機能は、ファイルを所有するノードが同時間帯に稼動するすべてのノードにファイル転送を行った。

- 実験内容

a 予想される稼働時間中にバックアップ/リストアを行う

b 予想される稼働時間以外でのバックアップを行い、予想される稼働時間中でのリストアを行う

c 予想される稼働時間中にバックアップを行い、予想される稼働時間以外でのリストアを行う

d 予想される稼働時間以外でのバックアップ/リストアを行う

稼働予想の信用性に対する本システムの耐性を調べるために、稼働予想が100%信用できる、80%信用できる、60%信用できる、40%信用できる、20%信用できると変化させながらリストアできるかを調べた。それぞれの場合についての実験を5回ずつ行い、リストア要求が出されていから時間と転送処理回数のそれぞれの数値と平均値を出力する。

自分以外に稼動しているノードがなく、分割されたバックアップファイルがすべて手元にある場合は除外した。1回ごとにバックアップファイル、転送リストなどを削除し、初期状態 (各ノードの共有ディレクトリに何もない状態) で、実験を行った。

6.3 実験結果

第 6.2 節で行った実験結果を表 6.1、表 6.2、表 6.3、表 6.4 に示す。表 6.1、表 6.2 の中の実験内容 b、d の転送回数は、() 前の数字は後転送の転送回数を示し、() 内の数字は前転送の転送回数を示す。

表 6.1: ノード条件 $<1>$ での実験結果

$<1>$						
実験内容 – 稼働予想の信用度	リストアにかかる時間 – 転送回数					平均転送時間 – 平均転送回数
$<a-100\%>$	7 2	8 2	8 2	8 2	8 2	7.8 2
$<b-100\%>$	8 2(10)	8 2(10)	8 2(10)	8 2(10)	8 2(10)	8 2(10)
$<c-100\%>$	14404 2	14404 2	14404 2	14404 2	14404 2	14404 2
$<d-100\%>$	8 2(9)	8 2(9)	9 2(9)	8 2(9)	8 2(9)	8.2 2(9)
$<a-80\%>$	8 2	8 2	8 2	8 2	8 2	8 2
$<b-80\%>$	8 2(11)	8 2(11)	8 2(11)	8 2(11)	8 2(11)	8 2(11)
$<c-80\%>$	14404 2	14404 2	14404 2	14404 2	14404 2	14404 2
$<d-80\%>$	7208 2(9)	7209 2(9)	7208 2(9)	7208 2(9)	7208 2(9)	7208.2 2(9)
$<a-60\%>$	8 2	8 2	8 2	8 2	8 2	8 2
$<b-60\%>$	8 2(6)	8 2(6)	8 2(6)	8 2(6)	8 2(6)	8 2(6)
$<c-60\%>$	8 2	8 2	8 2	8 2	8 2	8 2
$<d-60\%>$	7208 2(8)	7209 2(8)	7208 2(8)	7208 2(8)	7208 2(8)	7208.2 2(8)

表 6.2: ノード条件 < 1 > での実験結果 (つづき)

< 1 >						
実験内容 – 稼働予想の信用度	リストアにかかる時間 – 転送回数					平均転送時間 – 平均転送回数
$< a - 40\% >$	8 2	9 2	8 2	8 2	8 2	8 2
$< b - 40\% >$	8 2(0)	8 2(0)	8 2(0)	8 2(0)	8 2(0)	8 2(0)
$< c - 40\% >$	7204 2	7205 2	7204 2	7204 2	7204 2	7204.2 2
$< d - 40\% >$	8 2(0)	8 2(0)	8 2(0)	8 2(0)	8 2(0)	8 2(0)
$< a - 20\% >$	8 2	8 2	8 2	9 2	8 2	8.2 2
$< b - 20\% >$	8 2(7)	8 2(7)	8 2(7)	8 2(7)	8 2(7)	8 2(7)
$< c - 20\% >$	14408 2	14409 2	14408 2	14409 2	14408 2	14408.4 2
$< d - 20\% >$	8 2(3)	8 2(3)	9 2(3)	8 2(3)	8 2(3)	8.2 2(3)
$< a - 0\% >$	8 2	9 2	8 2	9 2	8 2	8.4 2
$< b - 0\% >$	7208 2(0)	7208 2(0)	7208 2(0)	7208 2(0)	7208 2(0)	7208 2(0)
$< c - 0\% >$	3608 2	3608 2	3609 2	3608 2	3608 2	3608.2 2
$< d - 0\% >$	7204 2(3)	7204 2(3)	7204 2(3)	7204 2(3)	7204 2(3)	7204 2(3)

表 6.3: ノード条件 < 2 > での実験結果

< 2 >						
実験内容 – 稼働予想の信用度	リストアにかかる時間 – 転送回数					平均転送時間 – 平均転送回数
$< a - 100\% >$	7 2	8 2	8 2	8 2	8 2	7.8 2
$< b - 100\% >$	86408 15	86408 15	86409 15	86408 15	86408 15	86408.4 15
$< c - 100\% >$	14404 10	14404 10	14404 10	14404 10	14404 10	14404 10
$< d - 100\% >$	9 2	8 2	9 2	8 2	8 2	8.4 2
$< a - 80\% >$	8 2	8 2	8 2	8 2	8 2	8 2
$< b - 80\% >$	86405 16	86404 16	86404 16	86404 16	86404 16	86404.2 16
$< c - 80\% >$	14404 12	14404 12	14404 12	14404 12	14404 12	14404 12
$< d - 80\% >$	8 2	7204 2	7208 2	9 2	7204 2	4326.6 2
$< a - 60\% >$	9 2	8 2	8 2	8 2	8 2	8.2 2
$< b - 60\% >$	8 2	10808 11	9 2	10808 11	10808 11	6488.8 7.4
$< c - 60\% >$	8 2	8 2	8 2	8 2	8 2	8 2
$< d - 60\% >$	7204 2	7208 2	8 2	8 2	7205 2	4326.6 2

表 6.4: ノード条件 < 2 > での実験結果 (つづき)

< 2 >						
実験内容 − 稼働予想の信用度	リストアにかかる時間 − 転送回数					平均転送時間 − 平均転送回数
$< a - 40\% >$	8	9	8	8	8	8.2
	2	2	2	2	2	2
$< b - 40\% >$	7204	7205	8	7024	7204	5765
	12	12	12	12	12	12
$< c - 40\% >$	7204	7205	8	7204	7204	5765
	2	2	2	2	2	2
$< d - 40\% >$	8	8	8	8	9	8.2
	2	2	2	2	2	2
$< a - 20\% >$	8	8	9	8	8	8.4
	2	2	2	2	2	2
$< b - 20\% >$	8	7204	7204	8	7204	4325.6
	2	11	11	2	11	7.4
$< c - 20\% >$	14404	14404	14404	14404	7204	12964
	5	5	5	5	2	4.4
$< d - 20\% >$	8	8	8	8	8	8
	2	2	2	2	2	2
$< a - 0\% >$	8	8	9	8	9	8.4
	2	2	2	2	2	2
$< b - 0\% >$	7204	7204	7204	7204	7205	7204.2
	2	2	2	2	2	2
$< c - 0\% >$	8	3604	3604	8	3604	2165.6
	2	2	2	2	2	2
$< d - 0\% >$	8	8	8	8	8	8
	2	2	2	2	2	2

6.4 考察

第 6.3 節での実験結果から考察を行う。以下に示すように実験内容別に考察を行った。

a 予想される稼働時間中にバックアップ/リストアを行う

稼働予想の信用度を変化させても、ノードの条件 $<1>$、$<2>$ ともに瞬時にリストアできた。これは、同じ時間帯に稼働しているノード群にファイルを配置するためであると考えられる。

b 予想される稼働時間以外でのバックアップを行い、予想される稼働時間中でのリストアを行う

稼働予想の信用度にかかわらず、条件 $<2>$ より条件 $<1>$ のほうが早くリストアできた。これは、前転送によって稼働中のノード群にファイルが転送される結果だと考えられる。また、信用度が高い場合には全転送数でも条件 $<2>$ より条件 $<1>$ のほうが良い結果になった。これは、ノードをグループ化して転送を行うことが有効であると考えられる。

ノードの条件 $<1>$ に関しては、稼働予想の信用度を変化させてもあまり変わらない結果となった。その一方で、ノードの条件 $<2>$ に関しては、稼働予想の信用度が高い時より低いほうが良い結果となった。これは、図 6.1 で示すように、信用度が低くなることで、稼働している時間帯が偏在することなく全時間帯を通して稼働するからではないかと考えられる。

c 予想される稼働時間中にバックアップを行い、予想される稼働時間以外でのリストアを行う

稼働予想の信用度を変化させても、条件 $<1>$、$<2>$ ともにリストアに要する時間があまり変わらなかった。しかしながら、稼働予想の信用度が高い場合のファイルの転送回数は、条件 $<2>$ より条件 $<1>$ のほうが良い結

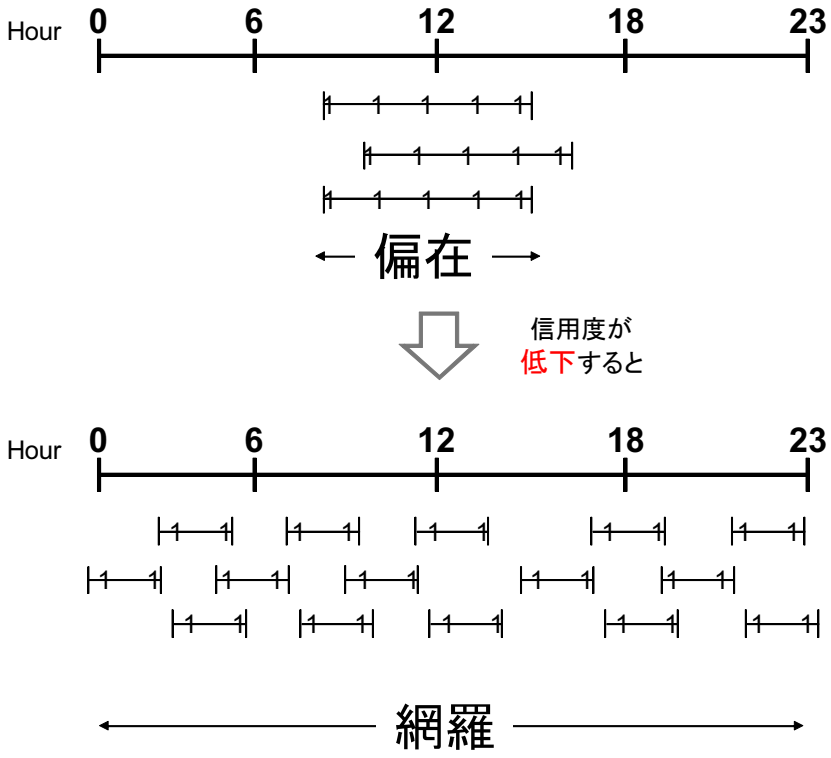

図 6.1: 稼働データの信用度低下による稼働時間の分散化

果になった。これは、ノードをグループ化して転送を行うことが有効であると考えられる。

d 予想される稼働時間以外でのバックアップ/リストアを行う

稼働予想の信用度にかかわらず、条件 $<1>$ より $<2>$ のほうが早くリストアでき、少ない転送回数でリストアできた。これは、ファイル転送を行うことで、バックアップを行ったノードと一番長い時間同時並行稼動するノードに転送されるためであると考えられる。

実験内容 d に関しては、条件 $<2>$ より条件 $<1>$ のほうが早くリストアでき、少ない転送回数でリストアできた。しかしながら、実験内容 d を除いた他の実験内容では、稼働予想の信用度が高い場合に、条件 $<2>$ より条件 $<1>$ のほうが良い結果であることが分かった。これは、稼働予想を使ったグループによるノードのファイル配置・転送が有用であった。

第 7 章

関連研究

個人用のコンピュータのハードディスクを利用して、ファイルの分散化、ファイルの保存を行っている研究は多数ある。代表的なものとして、P2P ファイル共有システムと P2P バックアップシステムが挙げられる。これらの大きな違いは、個人用のファイルをネットワーク全体で共有するか個人用のファイルとして扱うかである。そこで、実際に個人用のコンピュータのハードディスクを利用している研究・アプリケーションについて述べる。

7.1 P2P ファイル共有システム

P2P ファイル共有システムは、各個人のコンピュータのハードディスクを利用してファイル共有を行うシステムである。代表的なものとして Gnutella[11] と Winny[12] が挙げられる。以下にその 2 つのシステムについての特徴を述べる。

- Gnutella

  Gnutella は PureP2P ネットワークを構築してファイルの共有を行う。ノード同士だけでファイルの検索と転送を可能にしている。ノードが Gnutella ネットワークに参加するときには、すでに Gnutella ネットワークに参加している他のノードを 1 台見つけ、そこに接続する。検索要求を送信すると、その要求は Gnutella ネットワークに参加しているノードに伝播する。検索要求に一致するファイルを持つノードはこの検索要求に返信する。このようにして、

検索要求を送信したノードへ検索結果が次々と返信される。これらの検索結果より、実際のファイル転送が行われる。実際のファイル転送は、該当するファイルを所有するノードと検索要求を発行した 2 台のノード間で行われ、他のノードは基本的に関与しない。そのため、ファイルの所持ノードと転送ノードを容易に調べることができる。

- Winny

  Winny は PureP2P ネットワークを構築してファイル共有を行う。所持ファイルのリストなどの情報は利用者間をバケツリレー式に転送される。ユーザ ID などはなく、どのファイルがどこから送受信されているのか利用者はわからない。ただし、自分がダウンロードを指定したファイルの受信状況だけは知ることができる。ユーザを指定してメッセージを送信したり、共有ファイルのリストを見たり、ダウンロードを指定したりすることはできない。

  また、ダウンロード指定したファイルを直接受信せず、いったん第三者のコンピュータに送信させてから受信する転送機能がある。通信速度の速いユーザに積極的に転送させることにより、効率的にファイルの拡散が進む効果と、匿名性を高める効果がある。

P2P ファイル共有システムは一般的に次のような 3 つの特徴を持つ。

- 拡張性 (Scalability)

  ノードを増加させることにより、より多くのファイルを共有できるようにする。

- 匿名性 (Anonymity)

  誰がファイルを所有しているかの調査を困難にする。

- 追跡不可能性 (Untraceability)

どのようにファイルが伝播したか、誰が最初にファイルを共有したかを追跡することを不可能にする。

P2P ファイル共有システムは、上記の 3 つの特徴を持ち、ファイルを共有することに重点を置いている。本システムは、P2P バックアップシステムであるため、確実にバックアップ/リストアができるかに重点を置いている。この点が、本システムと P2P ファイル共有システムとの性質の違いである。

7.2 P2P バックアップシステム

- pStore: A Secure Peer-to-Peer Backup System

  pStore[16] は、P2P ネットワークを使った分散バックアップシステムである。バックアップデータに対して信頼性と分散冗長性を提供するため、インターネットに接続している個人の空きハードディスクスペースを利用する。サーバのどれかに悪意があるか、またはサーバが利用できない場合に備えて、データの複製をいくつかのノードに配置することでデータの信頼性を保っている。また、カスタムバージョンシステムは、CVS に類似した任意のバージョン復元機能を備えている。プライベートデータは所有者だけが読めるようにし、所有者だけが削除できるようにしている。この削除は、遠隔的に行える。

  pStore は、信頼性、セキュリティ、資源効率に重点を置いて研究を進めているが、本システムはリストアにかかる時間短縮とファイル転送コストの削減に重点を置いている。そのため、本システムと pStore とは目的が異なる。

第 8 章

問題点と課題

本研究では、生活リズムから求められる稼動予想に基く P2P バックアップシステムを提案した。本章では、本システムを使う上で起こり得る問題点について述べ、さらに今後の課題について述べる。

8.1 セキュリティ問題

本システムを使用するにあたり、セキュリティにおける問題点が挙げられる。

- 転送リストの改ざん

  転送リストを改ざんすることにより、ファイルが正しい目的ノードに転送されなかったり、本来転送されるはずのないファイルが他のノードに転送される。また、ある特定のノードにだけファイルが集まるような転送リストを生成され、ネットワーク上に流されることが考えられる。

  これを解決するためには、転送リストがシステムでしか分からないように暗号化を行うようにする。暗号化を行うことで、たとえ改ざんされたとしても、改ざん後のファイルはシステムで読み込むことができなくなるため、上で述べた問題は起こり得にくくなる。

- 共有ディレクトリに保存されているバックアップファイルの改ざん、削除またはバックアップファイル名の削除

ノードが保持している共有ディレクトリの内部のファイルデータまたはバックアップファイル名の改ざん、削除される可能性が考えられる。これによりバックアップファイルを正しく復元できない可能性が生じる。

これを解決するには、複製数を増やすことで対応できると考えられる。しかしながら、ディスク容量とのトレードオフが発生するため、使用規模などそのシステムにあった複製数を決める必要がある。

8.2 使用ディスクの均一化

グループ化することにより、バックアップを行う頻度によってノード間でディスク使用率に差が生じる。バックアップを頻繁に取るノード群では、各ノード内のディスク容量を使い切る可能性がある。このような状況下では、バックアップやファイル転送を行うことが出来なくなるため、他のノードにファイルを転送する必要がある。しかしながら、他のノードにファイルを転送する場合、複数あるバックアップファイルのうち、どのバックアップファイルを転送するかが問題となる。そこで、この問題を解決するために、バックアップファイルの重要度を設ける。ファイルの重要度を利用することで、複数あるバックアップファイルから適切なバックアップファイルを選択し、ファイル転送を行うことでそのノードのディスク使用率を下げるのがよいのではないかと考えられる。

8.3 稼働周期の変化への対応

人間は永久的に同じ生活リズムで生活することはなく、大多数の人間はある程度の期間で生活リズムが変化する。しかしながら、本システムに組み込んだ自己相関係数だけでは、稼動周期の変化には対応できない。そこで、この問題を解決するためには、短期的な予測と長期的な予測を組み合わせた方法を使う。また、最近の

稼動データに重みをつけることで、すばやく変化に対応でき、長期的なデータを用いることで予測が外れたデータが無効なデータかどうかを判別できると考える。

第 9 章

おわりに

本研究では、人間の生活リズムから得られる稼動予想に基いたP2P を使ったファイルのバックアップシステムの設計、実装を行った。P2P ネットワーク上でのファイルの送受信を行う際に、ノードの稼働状況によってファイルのリストアに影響するという問題点を解決するために、人間の生活リズムからノードの稼動周期を調べ、その稼動周期より稼動予想に基いたファイルの配置・転送を提案した。

ノードの稼動予想を求めるために、ノードの稼働状況を自動的に取得し、曜日毎で自己相関係数を適用した。これにより、そのノードの稼動周期を求めることができ、その稼動周期より稼動予想を求め、ファイル配置やファイル転送を行う際の情報とした。

より少ないノードへのファイル転送を実現するために、稼動予想から一番長く同時並行的に稼動するノード群にファイルを転送することにした。これによりファイル転送を行うノード数を減らすことにした。また、前転送処理を行うことにより本来の稼働中での比較的早いファイルの復元を実現した。

本システムで実装した稼働予想によるノードのグループ化アルゴリズムの有用性を調べるために実験を行った。結果で示されるとおり、これにより稼動周期を用いない方法より短時間でのリストアやより少ない転送処理でバックアップファイルをリストアできることを示した。また、P2P バックアップシステムで人間の生活リズムを考慮したノードのグループ化アルゴリズムは有用であることが分かった。

謝辞

本研究を遂行するにあたっては、いろいろな方々にお世話になりました。

まず、指導教員の多田好克先生には日頃から熱心なご指導、そしてご鞭撻を賜りました。さらに本研究のシステムについて、発想の段階からお付き合いいただきました。また、ご多忙中にもかかわらず論文の草稿を丁寧に読んで下さり、大変貴重なご助言をいただきました。

また、佐藤喬助教には研究を進めるにあたり、貴重な助言をいただきました。ここに厚く御礼申し上げます。

そして、本研究が行なえたことは 、研究方針や方法論について議論をし 、共に研究生活をおくってきた多田研、小宮研、そして村山研の学生諸氏のおかげでもあります。

最後に、今まで育ててくださった両親そして家族に深く感謝いたします。

参考文献

[1] Microsoft Windows XP,
http://www.microsoft.com/japan/windowsxp/default.mspx

[2] Microsoft Windows Vista,
http://www.microsoft.com/japan/windows/products/windowsvista/default.mspx

[3] filebank, http://www.filebank.co.jp/

[4] マイキャビ, http://www.nifty.com/cabinet/

[5] Yahoo! JAPAN, http://www.yahoo.co.jp/

[6] 渡邊恵太, 安村通晃 : “ReflectiveVision:「記録-再生」循環による日常生活リズム形成システムの提案と試作”, The 21st Annual Conference of the Japanese Society for Artificial Intelligence, 2007.

[7] 小松正史 : “地域社会の時間性 〜自然的時間から組織的時間へ〜”, 明治大学大学院農学研究科 1995 年度修士論文, Mar., 1996.
http://www.nekomatsu.net/results/paper_pdf/komatsu_meiji96_masterpaper.pdf

[8] 総務省 : “ネットワークと国民生活に関する調査 報告書”, Mar., 2005.
http://www.johotsusintokei.soumu.go.jp/linkdata/nwlife/050627_all.pdf

[9] 石村貞夫 : “SPSS による時系列分析の手順”, 東京図書, ISBN4-489-00727-2, Sep., 1999.

[10] 中塚利直 : “時系列解析の数学的基礎”, 教育出版, Nov., 1978.

[11] Gnutella Protocol Development,
http://rfc-gnutella.sourceforge.net/index.html

[12] 金子勇：“Winny の技術”, ASCII, ISBN4-7561-4578-5, Oct., 2005.

[13] Roger Dingledine, Michael J. Freedman , and David Molnar : “The free haven project: distributed anonymous storage service,” *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability*, ACM, pp.67-95, 2001.

[14] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan : “Chord: A scalable peer-to-peer lookup service for internet applications,” *ACM SIGCOMM Computer Communication Review 31 (4)*, pp.149-160, 2001.

[15] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong4 : “Freenet: A distributed anonymous information storage and retrievalsystem,” *In Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, pp.46-66, June, 2000.

[16] C. Batten, K. Barr, A. Saraf, S. Treptin : “pStore: A secure peer-to-peer backup system,” *MIT Laboratory for Computer Science*, Dec., 2001.