マルチメディア通信と分散処理 74－5
グループウェア 15－5
（1996. 1. 25）

LANにおける自律管理システムに関する研究: ユーザー管理と共有ファイルシステム管理

宮本幸彦 伊東靖英 山下雅史
広島大学工学部

近年増加するシステム管理者の負担の軽減を目的とした自律管理システムの研究について報告する。

本研究では、既に提案したユーザー管理と共有ファイルシステム管理に着目しており、ディスク管理という相互の問題をファイル移住を用いて問題解決し、統合システムの提案を行なった。クラスタ化された相互独立なネットワークにおいて自律分散型モジュールが機能する。ファイル移住を必要とする状況を定義し、各ファイルサーバに保持させる情報を説明した後、ファイル移住を円滑かつ効果的に行なうために各モジュールが用いる各種アルゴリズムの概略を示す。そして、最後に管理者からの情報を必要とする処理の実現方法について説明する。

A Study on Autonomous Administration System for a Local Area Network: User Administration and Shared File System

Yukihiko Miyamoto, Yasuhide Ito, Masafumi Yamashita
Fuculty of Engineering, Hiroshima Univ.

This paper is to report some accomplishments by a study on the Autonomous Adiministration System for the purpose of decreasing the burden to the system adiministrator.

Disk Management of a relative problem between User Administration and Shared File System is solved by using File Migration. As a result, it constructs the integrated administration system. In a clustering network, autonomous distributed modules function. First events which need migrations are defined and what information a server has are explained. Second algorithms for smooth and effective migrations are indicated. Finally how to realize managements occured by requests from the system administrator is explained.

1 はじめに

近年押し寄せるダウンサイジングの波によりネットワーク計算機環境はこれまでの集中型の環境から分散型の環境へと変化しつつある．そのためにこれまで特定の計算機のみを管理すれば十分であった管理者が複数の計算機を同時に管理しなければならなくなってきている．このことによる管理者の負担はネットワーク規模の拡大に伴い増大する一方である．この負担を解消もしくは軽減するためにはシステム管理作業を代替してくれる管理システムが必要である．また全ての作業を行なうためには管理者から自律したものでなければならず，さらにそのネットワーク規模に左右されないために分散型とすべきである．つまり，自律分散型のシステムである．ここでは自律管理システムと呼ぶ．

そのシステム管理はルーティング管理，ユーザー管理，ファイルシステム管理，セキュリティ管理，デバイス管理など様々なものからなる．しかもこれらの管理作業間には相互関連する点が多々存在する．我々の求める自律管理システムはそれらの関連箇所を含めた全てのシステム管理を統合することによって構築される．

本稿では，これまでに我々が各個に支援システムを提案しているユーザー管理およびファイルシステム管理の統合システムの提案を行なう．そのシステム構築に際して特に両者に相互関連するため問題点として残されているディスク管理に着目し，その解決策としてファイル移住という手法を用いた．比較的規模の大きなネットワークを対象としその規模に左右されないよう分散型モジュールを用いるシステムとした．各モジュールは同一のアルゴリズムに基づき他モジュールの欠落等の障害にも左右されずに機能する．また負担軽減の目的からも管理者からの情報提供を最小限に押えるために各モジュールがネットワークシステムにおける環境維持という処理を自己の判断で行なうよう自律的に動作させる．

2 統合システム

2.1 モデル

UNIX システムのマシンを有する局所ネットワーク (LAN) 上においてルータにより分断されるサブネットワークを cluster とする．この cluster の一つを管理対象ネットワークとする.(図 1) cluster 内のマシン数は数台〜数百台，サーバ数は数台〜数十台存在するものと想定する．

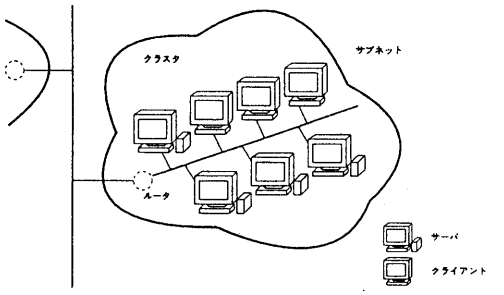

図 1: 管理ネットワーク

LAN 内のディスクは論理パーティションがなされており，cluster 内のディスク集合はこの論理パーティションの一分割に相当する．ネットワークシステムは各 cluster 内のディスクを一つのディスクとして認識する．分散ファイルシステムによりファイル共有が行なわれておりファイルシステムを cluster 内でのみ共有可能である．

ユーザーのホームディレクトリは cluster 内に唯一存在するが cluster 内のディスク上に分散することを認める．各 cluster は互いに独立であり，cluster 内でのユーザー識別子は一意である．

さらに以下のことを仮定する．書き込みループ状態に陥りファイルシステムに多大な影響を及ぼす強制停止が必要なプロセスは存在しない．cluster 内のディスク上には必ず空き領域が存在する．ディスク上に存在するファイルシステムは一つである．

2.2 問題点と解決策

自律管理システムを構築する先駆けとして，本研究ではユーザー管理とファイルシステム管理を統合したシステムを提案する．この両者を統合するためには両管理作業間に存在する問題点であるディスクの空き領域の管理を行なわなければならない．管理者への負担を減らすためにシステムはファイルシス

テムを常に利用可能な状態に保つ作業を行なわなければならない．そのため以下の処理を行なう．

- ファイルシステムがフルになった場合に他のファイルシステムへファイル移住 (*File Migration*) を行ない，一時的に作業環境の回復を行なう．
- ファイルシステムがフルになることを予想しファイルを移住させ空き領域を確保する．
- ユーザー登録時のホームディレクトリの物理的位置はシステムが決定する．
- クライアントが同一サーバからファイルシステムをマウントすることでその構成が複雑になることを避け，移住によるサーバとネットワークの負荷を軽減する．
- ユーザーの利用環境が変化させず一定に保つ．

2.3 Events

先の理由からディスク不足と増設/撤去に対してディスク間のファイル移住により一定の空き領域を確保しディスク維持を行なう．空き領域の不足を含めた移住を要する状況を Event と呼び，発生周期によって，三種類に分類する．(表 1を参照.)

表 1: *Event*

周期	要因
随時	ホーム作成/削除，不慮の書き込み
短期	フルの予測，ディスク撤去，無条件†
長期	マウント構成の複雑化

†ディスクフルの予測とは異なるアルゴリズムを周期毎に無条件に実施する．

サーバはファイル移住を要する複数の Event を同時に処理する必要はない．ファイル移住を並列に処理することはサーバの負担になる上，移住が完了する保証はなく未完の場合には他のサーバを選択しなければならないので時間を浪費することになる．そこで，各サーバでの Event の発生は一つのみとする．但し，メッセージ伝搬についてはこの次第ではない．

本システムでは既存のアプリケーションを利用することを念頭に置く．

2.4 隣接リスト

ファイル移住を効率的かつ耐故障性を持つものにするために移住可能と思われるディスクを有するサーバ集合を各サーバに保持させ，その情報を用いたアルゴリズムを実行させる．そのサーバ集合を隣接サーバ，隣接サーバを要素に持つリストを隣接リストと呼ぶことにする．アルゴリズム説明の簡単のために各サーバの隣接リストから，サーバが頂点であり自身を始点とし隣接サーバを終点とする有向辺によって構成される有向グラフを用いる．グラフ構成例を図 2 に示す．このグラフは必ずしも強連結グラフではなく完全独立なグラフの集合であっても良い．また新たなサーバは隣接リストに要素を持たずにグラフにおいて単独頂点の形で追加される．その他に予測と移住先サーバ選択のための情報を蓄える用意をする．サーバは各環境に適応した処理を行なうためにこれらの各種情報を随時更新していく．

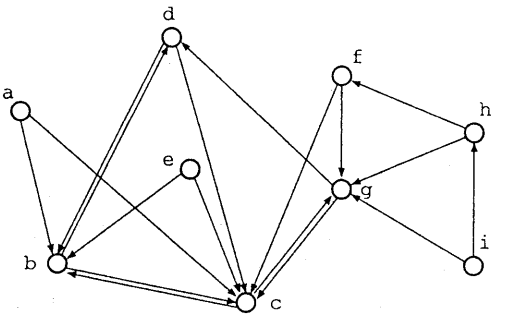

図 2: グラフ構成例

2.5 移住情報

ディスク使用量の増減のデータベースを各サーバ毎に用意し，向う数週間の使用量の平均値により短周期の Event の発生を予測する．使用頻度の傾向がその利用時期に反映される可能性を示唆し，この使用量は一昨年までの同時期の情報を用いる．但し，その情報が蓄えられていない場合には代わりとして過去数週間の使用量の平均を用いることとする．

またファイル移住を行なった際の受け入れを正，放出を負として過去数回の移住ファイルサイズの総量を求める．その和が正 ($\geqq 0$) であれば *in dominant*,

負であれば *out dominant* とし隣接リストの再構成に用いる. これは過去の移住傾向からディスクが今後取り得る状態の予測を行なうことを意味する.

サーバはファイル移住の受け入れ自体が自身の負担となるので移住させたサーバに対して代償を求めるべきである. そこで移住したファイルのサイズをその移住元サーバと共に保持させ, 受け入れたサーバが他サーバへの移住を行なう際にサーバ選択要因の一つとする. ファイルサイズに関しては受け入れた量を正とする.

また, ファイルの分散を防ぐために同一サーバへの移住を優先させる. このために前移住先サーバを保持し, 次回のサーバ選択時に優先的に選択させる.

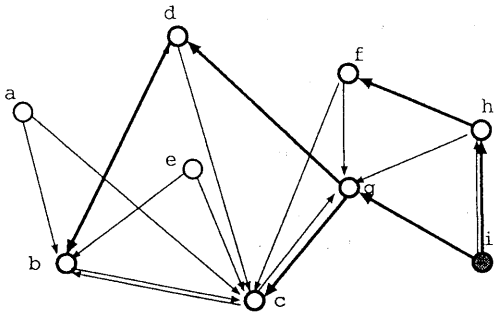

図 3: メッセージ伝搬 (*occurrence* i)

3 アルゴリズム

3.1 メッセージ伝搬

各サーバは随時および短周期の Event の発生時には各サーバへその発生を伝達しなければならない.

アルゴリズムは効率化のために Event の発生したサーバを根とする木を形成しつつ並列にメッセージの伝搬を行なうものになっている. また隣接サーバの故障あるいはネットワークの輻輳によりサーバ間で通信不能な状況を想定し, 通信不能なサーバを回避してメッセージ伝搬が行なえるよう隣接リストの要素数をある一定数以上とする. アルゴリズムについてはその詳細を省略するが, 各サーバが自身の隣接サーバへメッセージを伝搬することで結果的に到達可能な全てのサーバへの伝搬が行なわれる. 但し,Event 発生サーバ (occurence) が全ての隣接サーバと通信不能な場合には隣接リストを再構成 (新たなサーバを付加) して伝搬を行なう. (第 3.5節参照)

図 2 のグラフ上での例を図 3 に示す.

3.2 ファイル移住

既存のアプリケーションソフト上での実現を考えるとファイル移住時には, ファイルをロックする方法を採らざるを得ない. 一定サイズ以上のファイルを各個に移住させ, 一定サイズより小さいファイルをそれを越えない複数のファイルでまとめて移住させる分割移住を行なう. 各ファイルブロック毎にファイルロックを行なうことで特定のファイルの長時間のロックを防ぐ. 但しユーザーのファイルアクセスを考慮する必要のないホームディレクトリ作成では一括移住 (作成) を行なう. この場合のファイルブロックは全ホームディレクトリである.

1. ファイル送信が不可能な場合にそれがファイルブロックの
   (a) 転送前であれば, そのファイルブロックの移住を終了しそのサーバへの移住も終了する. 必要ならば他のサーバへ移住する.
   (b) 転送中であれば, 移住途中のブロックを削除しそのサーバへの移住を終了. 必要ならば他のサーバへの移住を行なう. 転送不能な状況としてはファイルシステムがフルになったことが考えられる. この時他の Event が発生するがすでにファイル移住を行なうべき Event が発生しているため無視される.(第 2.3節参照)
2. ファイル移住が全く不可能であった場合
   (a) その Event がファイルシステムフルによるものであれば隣接リストを再構成し再度移住を行なう.
3. 完全にはできなかった場合にその Event が
   (a) ファイルシステムフルか簡単化によるものであればそのまま終了する.
   (b) ホーム作成であれば再構成し再度移住を行なう. これは全ファイル移住が完了するまで繰り返される.

(c) 予想であれば再構成後終了する.

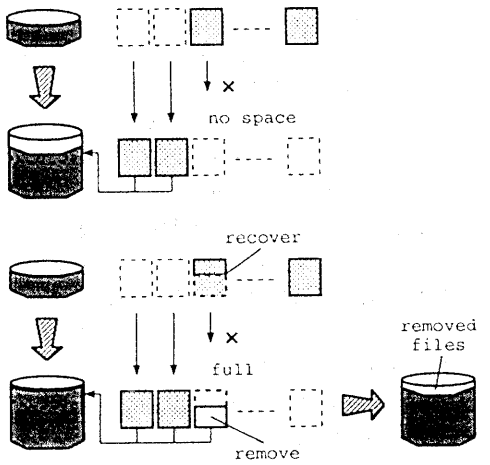

図 4: 移住中エラー処理

移住ファイルの選択については, 予測に基づく移住のみ以下のように選択する. 他サーバからの移住ファイルを保持している場合にはそれらを優先し, それ以降は順次ラストアクセスの古いもの, ホームディレクトリを全て移住できるものと選択する.

3.3 移住サーバの選択

随時の Event では, メッセージ伝搬後の最初に応答のあったサーバをその移住先とし, 緊急時の一時救済処置としての移住を行なう. 実質的な移住は短周期の Event に任せるものとする.

短周期の Event では, 一定期間内に応答のあったサーバの中から前移住先サーバ, 以前移住を受け入れたサーバで最もその量が多いもの, そして空き領域の多いものという優先度で移住先サーバを選択する.

3.4 マウント構成の簡単化

クライアントが同一サーバからマウントすることでサーバとネットワークの負荷を軽減させるために複雑になるマウント構成を簡単化する. 長周期の Event に対して以下のような処理を行なう.

仮定として, ファイルが元々配置されていたサーバは自身がファイル配置の起点となったサーバであることを知っている. このサーバを起点サーバと呼ぶことにする. マウントされたファイルからクライアントはファイルの分散配置先を知ることができ, 各サーバはそのクライアントを知っている.

1. *occurence* は自身の持つディスクの空き領域から移住可能な領域を算出し, 分散したファイルのマウント設定がなされているクライアントの一つへその情報を含むメッセージを送信する.

2. クライアントは分散している各ファイルブロックの中から *occurence* へ移住可能なものを選ぶ. 移住可能なファイルが存在すればそのファイルを所有するサーバを含んだメッセージで応答する. 但し, 起点サーバは移住の対象としない.

3.5 隣接リスト再構成

再構成は, ファイル移住後と移住先が特定できなかった場合及び短周期毎に行なう. 再構成後の隣接リストを用いたメッセージ伝搬が移住を受け入れる可能性のより大きな *in dominant* なサーバに対して行なわれるために隣接リストの要素を *in dominant* なサーバを多く含むものへと再構成すべきである.

1. ファイル移住後,

   (a) *occurence* は最終的な移住先サーバに一定量以上の空き領域が残されていれば, その伝搬経路となった隣接サーバをこのサーバへ変更する. 但し, 随時 (フル) では適用しない.

   (b) 応答サーバの内 *out dominant* でない一定量以上の空き領域を持つものを伝搬経路となった隣接サーバの代わりに新たな隣接サーバとする. 但し, 経路となった隣接サーバ自身がその候補に含まれる場合は変更ではなく付加するものとする.

2. 移住先サーバが存在しない場合 (以下の状況),

   - *occurence* が新設サーバである
   - メッセージ伝搬後サーバからの応答が得られなかった
   - 移住時に応答サーバ内に移住可能なサーバが存在しなかった

occurence からメッセージを受信していない cluster 内のサーバから一定量以上の空き領域を持つサーバの一つを新たに隣接リストに加える. この時同一の *occurence* とリクエスト番号を用いることで確実にメッセージを受信していないサーバかもしくは他の独立なサーバ集合との連結性を確保できる.(図 5)

3. 短周期毎

   (a) 隣接リスト上のサーバ数が最小辺数でない限り *out dominant* で最大の値を持つサーバを削除する.

   (b) *in dominant* に切り替わった頂点はその次の周期の再構成において隣接サーバへメッセージを伝達し各サーバの隣接リストに加えられる.

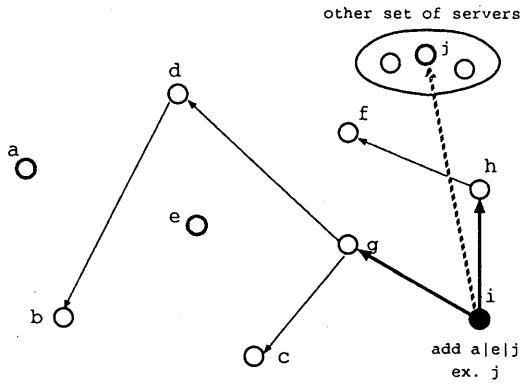

図 5: 再構成例 (他サーバ集合との連結)

4 管理作業の実現

管理者の行なうユーザー登録・削除とディスク増設・撤去・移設という処理の実現について説明する.

1) ユーザー登録時には管理者から要求を受けたユーザー管理支援システムが本システムのモジュールにホームディレクトリ作成の要求を送りつける. それを受けて, ホームディレクトリが疑似的な一定の大きさを持って作成され移住先サーバでは元のサーバから既存のファイルが移住させられたように振舞うことになる. 2) ユーザー削除時ではホームディレクトリを保存しておく可能性もあるので削除の可否を確認する. 削除の場合にはファイル移住の対象となるファイルをそのユーザーのホームディレクトリとし全ファイルを, 存在しないサーバへ疑似的に移住させることで削除を実現する. 3) ディスクの増設についてはシステムへ委ねるファイルシステムを決定してシステムを立ち上げるのみで良い. システム自身には何の変化ももたらされない. 4) ディスク撤去では全ファイルシステムが他へ移住されるまでファイルシステムフルかフル予測のイベントを発生させる. 処理に必要な時間の関係上, 緊急時にはフルを用いさもなくば予測を用いる. 5) ディスク移設はシステムダウン後, そのまま新しいマシン上でシステムを立ちあげるのみで必要な情報が保存される. 但し他サーバからは新設サーバとみなされる.

5 おわりに

本研究では, ファイル移住という手法を用いたユーザー管理と共有ファイルシステム管理を統合した自律管理システム構築のための各種アルゴリズムの提案を行なった. これらのアルゴリズムから構築されるシステムでは管理者が直接の情報操作を行なうことも可能であるために完全に自律的なシステムとは言えないであろう. しかし本システム導入により管理者の負担を減少させることは可能である.

参考文献

[1] Swarup Acharya and Stanley B. Zdonik "An Efficient Scheme for Dynamic Data Replication," Dept. of Computer Science Brown Univerrsity Providence, RI 02912-1910, September, 1993.

[2] Curry,Kimery,De La Croix,and Schwab, "ACMAINT: An Account Creation and Maintenance System for Distributed UNIX Systems," Liza IV, Colo. Springs, Colorado, October 17-19, 1990, pp.1-10.