ファイルシステム共有環境における負荷分散方式の研究

村田 努　　井内 稔　山崎 晴明

山梨大学

概要

分散コンピューティング環境に接続されているマシンの利用状況は多様で、マシンの能力をフルに使っているものもあれば、アイドル状態のものもある。過負荷なマシン上のプロセスをアイドル状態のマシンに移して実行することができれば、ネットワーク上のコンピュータの負荷を分散し、処理の効率をあげることが可能と予想される。本稿では現行の OS のカーネルを変更せずに負荷分散を行なうための方法について論じる。

1　はじめに

近年、コンピュータの性能は飛躍的に向上しているにも関わらず、その価格は低下する一方である。また、ネットワーク技術についても同様に向上してきており、低コストでコンピュータネットワークを構築することが容易になってきている。そのため、既にコンピュータネットワークを社内あるいは学内に所有している企業・大学では拡充が盛んに行なわれ、それまで所有していなかったところでも新規に導入しやすい状況になってきた。

山梨大学に於いても平成 4年 3月、電子情報工学科 Iコースの教育用計算機システムが集中型システムから分散システムに変更になった。新旧の教育用計算機システムを利用してみた感じを比較してみると、集中型システムと分散型システムの違いが浮かび上がってくる。分散システムのほうが勝っている点としては

1. 耐故障性の向上
2. 利用するコンピュータを選択可能
3. 拡張性の良さ

などが挙げられる。しかし分散システムには欠点がないわけではない。ネットワークの性能・信頼性に依存している部分が多いため、ネットワークに障害が発生した場合、孤立してしまう可能性や、知らぬ間にパスワードなどの各種の情報を盗聴されている可能性もある。

*A Load Balancing Scheme on the Shared File System Environment.
by Tsutomu MURATA, Minoru IUCHI and Haruaki YAMAZAKI　(Yamanashi University)

分散システムにも欠点は存在するが、得られるサービスのことを考えれば非常に有益なツールであることは間違いない。しかし、現段階ではこれらのツールは本領を発揮できずにいる。それは分散環境での使用を前提に設計された、分散オペレーティングシステムが十分に普及してないことにも起因している。現状は従来から利用されている UNIX[1] オペレーティングシステムで、分散環境指向のアプリケーション (rlogin, rsh など) を利用しているだけに過ぎない。カーネル自体が分散環境向きには作られていないため、他のホスト上で動いているプロセスの動向を把握するようなことは得意ではない。もしそのようなことが可能になれば、ネットワークに接続されている各ホストのプロセスの状態、CPU の利用率などをリアルタイムに把握し、各ホストが負荷を分散するために協調し合うことも可能になる。

現在多くの企業・大学等で分散オペレーティングシステムの研究・開発が行なわれているが、実用システムとして世にでてくるまでには、もう少し時間がかかりそうである。そこで本稿では、既存の UNIX オペレーティングシステムで上述のような協調動作を実現するためにはどのような問題点を解決しなければならないか考察する。更に、その考察に基づいて実際に UNIX オペレーティングシステム上で動作する負荷分散機構を設計・実装する。

[1]UNIX は AT&T の登録商標です。

2 負荷分散の有用性

今日では、コンピュータは専門家だけが使用する機器ではなくなり、誰もが使用する道具となってきている。初めて使用したコンピュータは分散システムだった、という人々も既に存在しているのではないだろうか。そして「分散システムというくらいだから、ネットワークに接続されているコンピュータが協同で作業をしているに違いない」と信じている人も少なからず存在するだろう。この考えは決して間違ってはいない。しかし、現実には図 1のように、数台

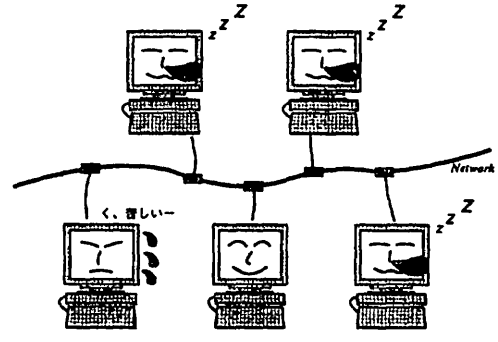

図 1: 各コンピュータの負荷の状態

のコンピュータのみが使用され、他のコンピュータはアイドル状態であることのほうが多い。このようなアンバランスを解決するために負荷分散を行なうことが望ましい。負荷分散を行ない、アイドルなコンピュータを最大限有効に活用することにより、

1. コンピュータが遊んでいる時間を減らすことができる。価格が低下してきているとは言っても、分散システムの構築にかかるコストを考えれば、高額な

買いものであることには違わないので、利用率が高いことが望ましい。

2. 過負荷なコンピュータの負荷を緩和させることができる。負荷が軽くなってくれば応答性が良くなり、ユーザのいらだちを軽減することができる。コンピュータを利用する作業に限らず、気分によって作業能率が左右されることは多いので、応答性の向上は非常に重要である。

などの利点がある。

3 既存の分散システムで負荷分散を実現する時の問題点

今日までに開発されてきた分散オペレーティングシステムの多くは負荷分散機能をサポートしている。これらのオペレーティングシステムは、分散システムでの使用を意識してカーネル等が設計されているため、負荷分散機構の実装はオペレーティングシステム自体でサポートされている。では、もともと分散環境での使用を前提に設計されていないオペレーティングシステムで、負荷分散を行なうためにはどのような問題を解決しなければならないのだろうか。本節では、UNIX オペレーティングシステム上で負荷分散を行なうために解決しなければならない問題点を取り上げる。

負荷分散で最も大切なことは、各ホストの負荷のバランスを取り、CPU の使用効率を改善させることだけではなく、プロセスの移送が起きた場合でも、プロセス移送が行なわれなかった時と全く同じように処理が実行されなければならないということである。つまり、負荷分散の機能はユーザに対して、「負荷分散機能が働いているために処理が快適に行なえる」ということを意識させるのではなく、プロセスが移送されたことをユーザに意識させないようにしなければならない。

既存の UNIX オペレーティングシステムで負荷分散を行なう時の一番の障害は、如何にしてプロセスをリモートのホストに移送するかということである。一部の分散オペレーティングシステムの中には、ローカルホストで実行中のプロセスをリモートホストに移送できるものもあるが、UNIX オペレーティングシステムでこの機能を実現するためには、プロセススケジューリングの部分、つまりカーネル内部に手を入れなければならない。

プロセスの移送先を決定するためには、リモートのホストの負荷の状態が把握できていなければならない。情報の収集方法はいろいろ考えられるが、定期的に収集する場合、周期が短いとリモートのホストの状況を的確に把握することができるが、通信量が多くなってしまう。逆に長くした場合には、通信量を減らすことができるが、古い情報を使わなければならなくなり、的確な判断ができなくなってしまう。適当な周期で収集が行なわれなければ、処理のオーバーヘッドや役に立たない情報の利用により、かえって効率を下げてしまうことになりかねない。

この 2 つの項目以外にも、入出力がらみ

の問題点がある。移送したプロセスがファイルからの読み出しを行なうものであった場合、そのファイルも移送先になければ処理が続行できなくなってしまう。またファイルへの書きだしを行なう場合には、実行終了後に移送元のホストへ該当ファイルを転送する必要がある。

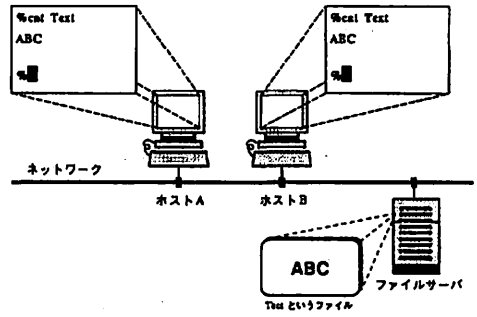

図 2: ファイルシステムの共有

4 負荷分散の実現

前節では現状の分散システムで負荷分散を行なう時に考慮しなければならない点についてふれた。本節では前節の問題点を踏まえて、既存の分散システムで負荷分散を実現するための方法について述べる。

4.1 対象システムの制限

既存のオペレーティングシステムを利用した分散システムについては様々な形態が考えられ、すべての形態に対し汎用的に用いることができる方法を考えることは恐らく不可能である。そこで負荷分散を実現させる対象を以下のような形態のシステムに制限することにより、既存のシステムでの実現を可能にする。

1. ファイルシステムを共有している: 分散システムを構築する際に利用できる手法の 1 つに、ファイルシステムの共有という方法がある。図 2にファイルシステムがマウントされているシステムの例を示す。図 2ではファイルサーバ、コンピュータ A, B が同一のネットワークに接続されており、ファイルサーバ上に Text というファイルが格納されている。2 台のコンピュータ A, B は共に Text というファイルをアクセスしている。**NFS**[2] (Network FIle System) を用いると、ユーザに対しリモートにあるファイルシステムを、あたかもローカルにあるかのように見せることが可能になる。

2. 同一の CPU アーキテクチャを使用: ネットワーク内に同一の CPU アーキテクチャを使用した機種が存在していること。**UNIX** オペレーティングシステム上で動作するソフトウェアは、ソースコードレベルで互換性があると一般に言われているが、バイナリレベルでの互換性はほとんどない。同一ベンダーが提供するコンピュータ間でさえ互換性がないことも少なくない。本研究では疑似的なプロセス移送を行なうため、移送先のホストが移送元のバイナリファイルを解釈できる必要がある。

[2]NFS は Sun microsystems 社の登録商標です。

3. どのホスト間の通信でも通信コストがほぼ一定: ネットワークに接続されているホストの中からどのような 1組を選んでも、自由に通信ができ、いかなる組合せを選んでも通信コストがほぼ等しいことが望ましい。

上記 3項目の制限は決して特殊なものではない。同一機種によって分散システムを構築し、ユーザのホームディレクトリを共有するような構成にすると、システム導入時に行なう設定作業のうち、各ホスト毎に設定しなければならない部分を除き、ほとんどの部分を共通化することが可能になる。システムの管理者の負担を軽減するために、このような構築方法はよく利用されている。

4.2 負荷分散機構の実現レベル

このような制限を設けてもクリアできない部分がある。それは実行中のプロセスをリモートのホストに移送することである。この機能を実現するためには、

1. 移送元ホストのカーネルの実行キューから移送対象のプロセスを取り去り、移送先ホストのカーネルに対し移送を依頼する。

2. 移送先ホストのカーネルが、移送されてくるプロセスのためのプロセス割当を自ホスト内で行なう。

3. 移送元ホストのカーネルは移送先ホストのカーネルにプロセスの状態、コードおよびデータなどを転送する。

4. 移送元ホストのカーネルは移送したプロセスのプロセス状態を削除し、移送先のホストのカーネルは移送されてきたプロセスの実行を再開する。

などの手順が必要になってくるため、カーネル内部のプロセススケジューリング機構の変更などが必要になってくる。カーネルの変更はリスクが大きいため、本研究では実行中のプロセスの移送は扱わないことにする。より多くのプロセスが移送の対象になることができ、なおかつカーネルの外で負荷分散を実現させるため、シェルに負荷分散機構を組み込むことにした。ユーザが

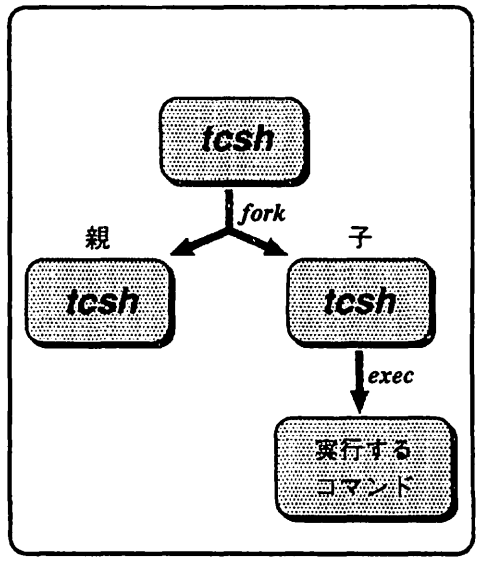

図 3: コマンドの実行

プログラムを実行する時、シェルは図 3に示すように自分自身を fork し、子プロセス側が exec を実行して目的のプログラムを実行する。このように、シェルで負荷分散機構をサポートすれば、ユーザが実行しよう

とするプログラムはすべて移送の対象になることができる。このような方法を採ることにより、なるべく多くのプロセスが移送の対象になることが可能になる (但し、負荷分散機構を組み込んだシェルを使用しないユーザのプロセスは対象にならない)。

4.3 負荷分散の方針

負荷分散の方針は「いつ、誰が、どのプロセスを、何処へ移送するのか? 」という設計思想の違いにより、様々なものに分類することができる。以下に本研究における方針を示す。

4.3.1 負荷情報の収集

本研究では CPU の使用状況を負荷の基準として用いる。リモートカーネルの統計情報を提供するデーモン (rstatd) は資源に関する様々な統計情報を提供しており、実行キューにある待ちプロセスの数、CPU の実行時間等の状況を知ることができる。各ホストは全ホストのデータを収集し、全ホストの負荷の状況を把握できるようにする。

4.3.2 移送の判定時期

移送の判定のタイミングは周期的に行なうものとイベント駆動型で行なうものに分けることができるが、本研究ではイベント駆動型の方法を用いる。負荷分散機構をシェルレベルで実現するので、シェルがプロセスを生成する直前 (図 3の中の exec を行なう直前) に判定を行なうことができる。

4.4 移送の主体

通常、分散システムを構築する際には、ピーク時にどの程度の利用があり、その時にどの程度のパフォーマンスがあれば良いかという基準で設計が行なわれるので、分散システムが常に過負荷な状態になっていることは少ないはずである (常に過負荷であるようであれば、それは負荷分散の方法を考える以前に、システムの設計を見直すべきである)。負荷の軽いホストが多いような環境では、負荷の重いホストが軽いホストへ移送要求を送る、送り手主導型のほうが移送要求が起こる可能性が低い。

4.4.1 移送するプロセス

本研究では実行中のプロセスの移送は行なわないので、新しく生成するプロセスを対象にする。具体的には、ユーザがコマンドを入力する度に移送すべきかどうか判定し、移送することになったらそのコマンドが移送される。

4.4.2 移送先の決定

本来は通信コストを考慮すべきであるが、LAN のような小規模なネットワークでは通信時間の違いは顕著に現れないので、負荷の軽いホストの中からどこか 1 つのホストを選択する。

5 負荷分散機構の設計と実装

前節で述べた方針に沿って負荷分散機構を作成し、山梨大学工学部電子情報工学科

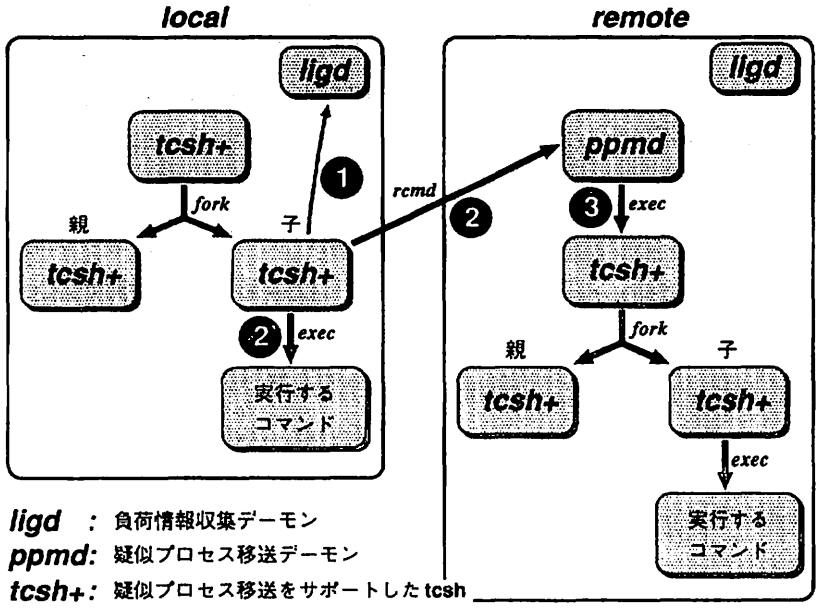

図 4: 実装した負荷分散機構

I コースの教育用計算機システムに実装した。本節では今回作成した負荷分散機構について説明する。

5.1 概要

本研究で作成した負荷分散機構は、リモートホストの負荷情報を収集し、適当な移送先を指示する負荷情報収集デーモン (ligd)、ローカルホストからコマンド、引数列、環境などを受けとる疑似プロセス移送デーモン (ppmd)、負荷分散機構をサポートするよう変更を加えたシェル (tcsh+) の 3 つの部分から構成されている。

5.2 負荷情報収集デーモン ··· ligd

負荷情報の収集を行なうデーモンで、前述の rstatd に問い合わせを行なうことにより情報を得ている。CPU の実行時間に関するデータから CPU の稼働状況を調べる。移送の条件を指定することができ、ローカルホスト、リモートホスト両方の閾値を設定することができる。移送はローカルホストの負荷が閾値を越え、かつリモートホストの負荷が閾値を越えていない場合にのみ起こる。移送を行なったほうが良い状況になった場合には、移送先が書かれたファイルを作成する。

5.3 疑似プロセス移送デーモン ··· ppmd

リモートログインデーモン (rlogind) に変更を加えたもので、移送元ホストからユーザ名・実行するコマンド・カレントディレクトリ・環境変数などを受けとる。端末モードの設定、移送先ホストで移送元ホストと

同等の環境の設定を行なった後、シェルを起動する。

5.4 負荷分散機構をサポートしたシェル

コーネル大学の Bob Byrnes 氏が作成した C シェル上位互換のシェル tcsh (version 6.03) に負荷分散機構を追加したシェルである。このシェルは目的のプログラムを exec する直前に ligd の作成するファイルの有無を確認することで疑似プロセス移送を行なうかどうか判断する。ファイルが作成されている場合、ファイルの中に書かれているホストが移送先になる。ファイルが作成されてない場合はローカルホスト内でコマンドを実行する。

6 評価

負荷分散の効果を確認するために、負荷のかかったホスト上でソーティングにかかる時間を計測した。

	負荷分散機構				
	なし		あり		
負荷	Real	User	Real	User	合計
0.0	27	23	27	23	-
1.0	52	23	28	24	31
2.0	73	23	28	23	31
3.0	97	23	28	24	31
4.0	120	23	28	23	32
5.0	148	24	29	24	32

負荷の欄の数字は測定用のプロセスを実行する直前に uptime コマンドで得られた数値で、表中の数字は実行にかかった時間(単位は秒)を表す。Real, User はそれぞれ実時間・プロセスの実行にかかったユーザ時間を表し、負荷分散機構を利用した方にだけ存在する合計の項は移送にかかった時間まで(実時間)を含めている。負荷分散機構を用いない場合、負荷のかかりぐあいによりほぼ線形に実行時間が増えていくのに対し、用いた場合には負荷がどんなにかかっていてもほとんど変わらない時間で実行できていることがわかる。またこの実験結果から、移送にかかるオーバーヘッドは約 3 秒程度(実時間)であることもわかる。

7 まとめ

本研究では、ファイルシステムが共有されている等の制約はあるものの、UNIX オペレーティングシステム上でも負荷分散を実現することが可能であることを実証した。

参考文献

[1] A. Goscinski, DISTRIBUTED OPERATING SYSTEMS The Logical Design, Addison-Wesley, 1991.

[2] 前川 守, 所 真理雄, 清水 謙多郎, 分散オペレーティングシステム UNIX の次にくるもの, 共立出版, 1991.

[3] W.Richard Stevens, UNIX NETWORK PROGRAMMING, Prentice Hall, 1990.
篠田陽一 (訳), UNIX ネットワークプログラミング, トッパン, 1992.