移動計算機環境におけるマイグレーションの一考察

6E－3

堀 正弘　伊藤 光恭
NTT(株) ソフトウェア研究所

1 はじめに

近年の計算機は高度にネットワーク(NW)化が進んでいる。今後はハードウェアの急速な進歩に伴って、高性能化小型化した携帯計算機が急増し、これらもNWに接続されて使用されるようになる[1]。

携帯用計算機は使用場所の制約が小さいため、トラヒックが、よりダイナミックに変化という性質が顕著になる。このため、NWを効率的に運用するためのトラフィック負荷分散の技術が更に重要になる。

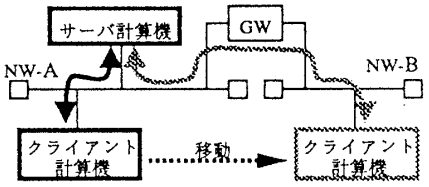

図1. 計算機の移動によるトラヒックの変化

一般に負荷分散は、ある計算機上で実行されている分散APの一部(プロセス)を他の計算機に移す(移送、或はマイグレーションと呼ばれる)ことによって実現される。プロセス移送に関する研究は、主に分散OSの機能の一つとして研究されてきた。しかし、その技術は未だ成熟しておらず[2]、既存のOSには適用できないこともあって、広く普及してはいない。

本研究では、現在広く用いられているUnix上でトラヒック負荷分散を実現するために、OS自体には手を加えない方法で移植性の高い移送機構の実現を目指し、RPCサーバ機能を移送するプロトタイプを検討した。

2 移送の対象と性質

2.1 RPCサーバの移送

移送の対象となる分散APのプロセス間通信は、RPC(Remote Procedure Call)によるものとした。これは、多くのNWワイドな計算機サービスがRPCのインターフェースを持つようになっており[3]、RPCベースのサービス提供形態が標準的になりつつあるからである。またこれは、ネットワークサービスのほとんどが、RPCを用いて記述することが可能なことを意味している。

従ってRPCのサーバプロセスの移送が実現できれば、多くのネットワークサービスに応用できるはずである。

2.2 移送に求められる性質

負荷分散を実現する道具としての移送に必要とされる性質を、(1)計算機NW環境の観点、(2)RPCサービスを受けるユーザの観点、(3)サーバ関数やクライアント関数を作成するプログラマの観点、の3つからまとめると、以下のようになる。

(1) 計算機NW環境の観点

(a) 何時でも実行できること。負荷分散を柔軟に行なうために必要。

(b) 異機種計算機、異なるOS間で実行可能。NWが全て同じOSの計算機で構成されることは、あり得ない。

(2) ユーザの観点
APの中断時間が無いか、非常に短い。つまり、高速であること。移送がユーザから完全に隠蔽されるのが望ましい。

(3) プログラマの観点
通常RPCと同様にプログラミングできる。従来のソースコードや手順に変更を加える必要がなければ、既存のソフトウェア資産が生かせるし、導入も容易である。

3 プロトタイプ

3.1 実現上の制約

前述の性質を満たすような移送機構のプロトタイプを作成中である。但し(1)(b)は、実現を簡単にするため目標から除いた。

プロトタイプは、以下の条件を満たすSun-OSのRPCのサーバ機能を、クライアントとのコネクションを見かけ上は切断することなく、他の計算機に移送する機能を持つ。即ち、一つのクライアントと複数のサーバで構成されること、サーバプロセスはファイルをアクセスしないことである。

これらの制約は、問題をできるだけ単純にするためのものである。

3.2 実現の方法

プロトタイプでは、プログラムコードとサービス関数中で用いられるデータの一部とを別々にコピーする方法をとっている。これは、RPCサービスを継続して実行できることを目標としたためである。

この方法では、移送前と移送後のプロセスは全く違うものであり、状態も一部しか保存されないので、一般的なプロセス移送と同じものではない。しかし、RPCサーバの機能を、別の計算機上で提供できるようにすると言う意味で、サーバプロセスの移送とは区別して、「サーバの移送」と呼ぶことにする。

3.3 実現のために必要な機能

Sun-RPC上でサーバの移送を実現するために、以下に挙げる機能が必要である。

(1) 移送要求を検出する機能

(2) 移送先ホストの決定と、そのための情報収集機能
NWの負荷を大局的に判断し、負荷の少ない計算機を探し出す機能と、そのための他計算機との通信機能。

'An Implementation of RPC Server Migrations in Mobile Computing Environments'

Masahiro HORI, Mitsutaka Ito, NTT Software Laboratories
1-9-1 Kohnan Minato-ku Tokyo 108 Japan

(1)と(2)については、プロトタイプでは未実現で、要求はコンソールから与えている。

(3) RPC の発行を停止する機能
移送作業中のコンシステンシを保つため、RPC 発行を禁止する機能。

(4) ポートマップエントリへのエントリ追加／削除機能
ポートマップは、TCP/IP プロトコルポート番号と、RPC プログラム番号との対応を取るためのテーブルである。サービスの禁止／登録のために、サーバクライアント間に一旦コネクションが確立した後にも、ポートマップを書き替える。

(5) プログラムと指定したデータを他の計算機にコピーする機能
プログラムコードと共に、サービスの実行状態(変数やプログラムカウンタ)をコピーする機能。サービス実行中の状態を他の計算機上に再現するために必要。
プロトタイプでは、プログラムファイル及び移送すべきデータをプログラマが陽に指定し、サーバが関数実行中は移送を禁止することによって、この機能を実現する。

(6) サーバプロセスを起動／終了させる機能

3.4 構成

プロトタイプの構成を図2に示す。3.3の機能のうち、(1),(2),(5),(6)を、各計算機上に常駐する移送制御デーモンが持ち、クライアント／サーバの各スタブ部分は、(3),(4)及びデーモンとの通信を分担する。

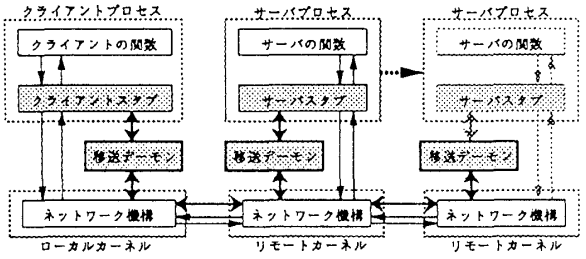

図2. 構成概要

3.5 移送の手順

サーバの移送は次のような手順で行なわれる。ここでは、サーバ移送元計算機、サーバ移送先計算機、クライアント計算機上のデーモンを各々、「元」「先」「ク」と表している。

(1) 「元」が移送要求を受けとる。

(2) 「元」は、クライアントが新たなRPCを発行しないようにするよう、「ク」に要求信号を送る。

(3) 「元」は、登録されたRPCのポートマップからエントリを削除して、サーバプロセスが新たなRPCを受け付けないようにする。

(4) 「元」は適切な移送先ホストを探す。

(5) 「元」は「先」に移送要求を送り、サーバプログラムと必要なデータを移送先ホストにコピーする。この時、移送先ホストで当該RPCと同じプログラム番号及びバージョン番号、またはソケット番号が既に使用されていれば、プログラムをコピーした後で変更する。

(6) コピーが完了したら、「先」はRPCサービスをポートマップに登録した後、サーバプログラムを再起動する。このとき、コピーしたデータを基に、移送前の状態を復元する。

(7) 「元」はサーバプロセスが占有していたソケットを解放し、サーバプロセスを終了させる。

(8) 「元」は「ク」に作業が終了した旨を通知し、新しいサーバのアドレス、ポート番号、プログラム番号、バージョン番号等々、必要な情報を渡す。

(9) 「ク」は、作業の終了通知を受けとると、使用中のポートを閉じ、クライアントを再バインドして、クライアントプロセスのRPCを許可する。

(10) RPCサービスが再開される。

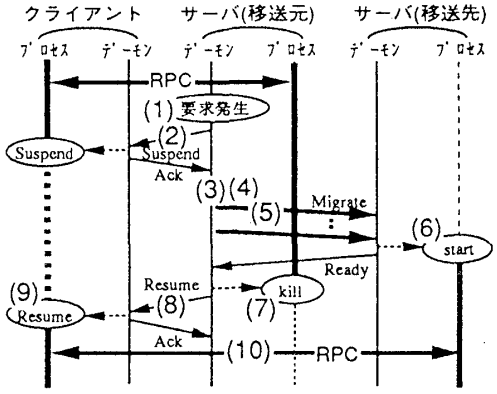

図3. 移送の手順

4 おわりに

特定の制約条件の下で、RPCサーバの移送を検討した。今後はプロトタイプを実装し、実行時間などを評価した後、異機種計算機間の移送や、制約条件を緩和した移送の仕組みを実現したい。またこの移送機能をCPU負荷分散にも適用していきたい。

参考文献

[1] T.Imielinski and B.R.Badrinath. *"Mobile wireless computing: Solutions and challenges in data managements"*. Technical Report, WINLAB/Rutgers Tech. Rept., February, 1993

[2] 前川守 所真理雄 清水謙多郎 編. "分散オペレーティングシステム UNIX の次にくるもの". 共立出版株式会社, 1991

[3] Object Management Group and X／Open, *"The Common Object Request Broker: Architecture and Specification"* , OMG Document No.91.12.1 (Revision 1.1), OMG, Framingham, Mass.,1991