3D-7

プロセス移送を前提とした入出力管理の枠組み

佐藤 友隆　　中山 健　　小林 良岳　　前川 守

電気通信大学大学院情報システム学研究科

1　はじめに

近年、個人でも複数の計算機をネットワークで接続して利用するようになってきた。また、計算機の小型化と移動通信技術の進歩によりモバイルコンピューティングも可能となってきている。これは、さまざまな場面において分散処理が可能になっている。このような状況におけるプロセス移送を考える。

従来、プロセス移送は負荷分散や耐故障性を主目的とする場合が多かったが、現在ではプログラムの移動を積極的に利用するモバイルエージェントも注目されている [1] [2]。

本研究ではプロセス移送時に生じるI/Oの切り替えに対応するためのOS側の機構を提案する。なお、本研究では個人の計算機環境を想定しているため、セキュリティに関しては考えない。

2　背景

分散処理においてプロセスの移動は欠かせない機能であるが、プロセス移送時のI/Oの取り扱いに関しては次のような問題がある。従来の目的でのプロセス移送は、移送した場合と移送しなかった場合とで、実行結果を同じにするため、移送後も移送元の計算機環境を扱っているように見せていた。このために、I/Oは透過的に移送元の計算機に転送する必要がある。一方、モバイルエージェントは移動先の計算機でI/Oを行うのが基本であるが、移動のたびにI/Oの切り替えが必要になる。

このようなI/Oの転送やつなぎ替えは、多くの場合OSレベルではサポートされておらず、ユーザレベルで実装されている。 [3] ではJavaを用いているが、Javaはプラットフォームに依存しない環境を提供することを目標としているため、I/Oに関しては抽象度が高くなっており小回りが効かない。そこで、単純で柔軟なOSレベルのI/O体系が必要になると思われる。

しかし、UNIX上ではI/Oの転送やつなぎ替えはもともと想定されていた機能ではないため、次のような問題が生じる。

- オープン中のデバイスは、そのままでは切り替えが困難
- リモート計算機のデバイスにアクセスできない。

本研究の類似の研究として [4] があるが、我々は特にI/Oに注目した研究を行った。

3　プロセス移送を前提としたI/Oの体系

本研究では、プロセス移送で必要となるI/O機能として、のフレームワークを提案し、オープン中のデバイスの動的な切り替えを考え、その機能をOSレベルのサービスとして提供するために必要となるOS内部の機構を提案する。

3.1　I/O切り替え

I/O切り替えは、あるデバイスへのアクセスを他のデバイスへ切り替える。

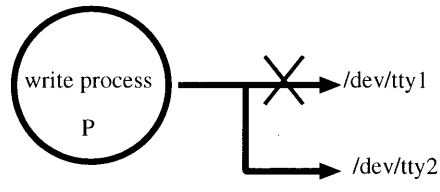

図 1　書き込みの切り替え

図 1では、プロセスPは/dev/tty1へ書き込みを行っているが、これをOSが/dev/tty2への書き込みに切り替えている。このときプロセスP側では切り替えを意識することはない。また、/dev/tty1へ書き込みを行っている途中であっても/dev/tty2に切り替える事も可能である。

プロセスPがデバイスからデータを読み込む場合も、切り替えが行われると実際にアクセスす

An I/O framework for process migration.

Tomotaka Sato, Ken Nakayama, Yoshitake Kobayashi and Mamoru Maekawa

Graduate School of Information Systems, University of Electro-Communications

1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan

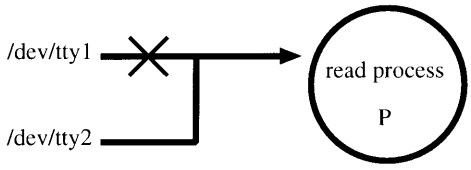

図 2 読み込みの切り替え

るデバイスは別のデバイスに変わる。

なお、切り替えを行ったときに、読み込みを行っていたプロセスは /dev/tty1 へのアクセスは不可能となる。これは、OS 側が /dev/tty1 へのアクセスを監視しており、アクセスがあるごとに /dev/tty2 へのアクセスに変換するからである。もし、/dev/tty1 に再度アクセスする場合には、OS 側が転送の解除をする必要がある。

3.2 リモートデバイスへのアクセス

通常、OS が直接扱うのはローカルなデバイスだけだが、これを拡張してリモートのデバイスも同様に扱えるようにする。

3.3 モデル

I/O 切り替えは次のような手順になる

1. 切り替えるデバイスを使用しているプロセスをサスペンドする
2. 切り替え元デバイスの状態を切り替え先デバイスに再現する
3. 切り替え元デバイスを使用していたプロセスを切り替え先デバイスを使用するように変更する
4. サスペンドしていたプロセスを再実行

ここで、切り替え先のデバイスの状態を、切り替え元の状態と等価にする必要がある。このためには、基本的にはデバイスドライバ内のデータを切り替え先にコピーすればよいが切り替え先のデバイスの種類が異なる場合は、単純には各デバイス固有の状態の再現は行えない。例えば、サウンドカードはその機能は同じだが、デバイスそのものは全く別のデバイスである。

そこで、デバイス固有のデータとそれ以外のデータに分離し、固有ではないデータを管理するレイヤをデバイスドライバよりも上位のレイヤに実現する。このレイヤで管理されるデータはデバイス固有ではないので I/O 切り替え時に状態の再現が行えることになる。

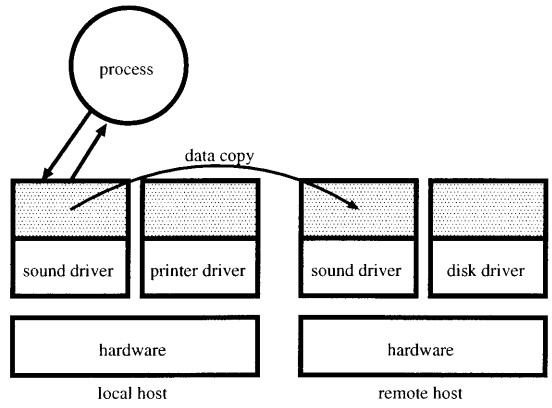

図 3 抽象化レイヤ

4 まとめ

本研究では、プロセス移送を前提とした OS レベルでの I/O の枠組みを提案した。今回提案した枠組みは一貫性のある体系として捉え直そうとしている。今後、この枠組みの実装と評価を行っていく。

参考文献

[1] Dejan S. Milojicic, Fred Douglis, Yves Paindaveine.: "Process Migration," TOG RI Technical Report, October 1996.

[2] David Chess, Colin Harrison, Aaron Kershenbaum.: "Mobile Agents: Are They a Good Idea?," IBM Research Report RC 19887, 1995.

[3] 村瀬, 若山, 権藤, 永田, 西尾, 徳田. "計算機環境に応じてサービス提供方式を適応させる通信用ツールキット" 情報処理学会研究報告, 2000-OS-84, 2000.

[4] 芝, 大久保. "分散オペレーティングシステム Solelc の構成". 情報処理学会研究報告, 2000-OS-84, 2000.