UNIXとWindowsNT 間のデータ共有方式

5Q－2

鶴 薫　細川 武彦
三菱電機(株) 情報技術総合研究所

1. はじめに

近年、コスト低減の市場要求により、従来 UNIX 系の OS を搭載したサーバー機、またはワークステーション(WS)を WindowsNT を搭載した標準的なパーソナルコンピュータ(PC)に置き換える動きが加速している。

本報告では、従来、複数台の UNIX サーバ機と WS で構成された分散型システムの一部を WindowsNT WS に置き換える際に発生する UNIX 上 S/W と WindowsNT 上 S/W 間でのデータ共有問題に対する低コストな解決手段を提案する。

2. 背景

産業用システムなどの特定分野においては、従来複数台の UNIX 系計算機を利用したシステム構築がなされており、図 1 に示すように計算機内及び計算機にまたがるプロセス間のデータ通信、及び、ファイルアクセスは、独自のミドルウェア(M/W)を介して行なっている。また、アプリケーションプログラム(App)においては、同一機種を前提として開発されていることから、ネットワークコード対応の処理が含まれていなかった。

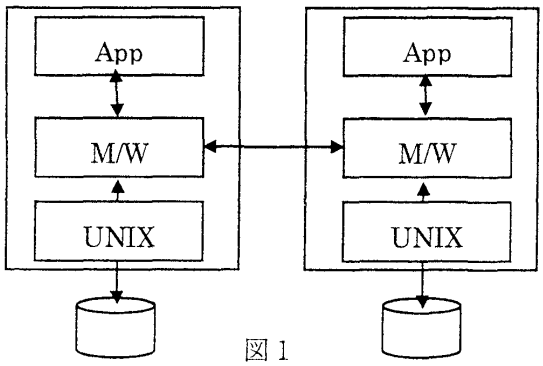

図 1

このような分散型システムにおいて、システムを構成する UNIX 系計算機の一部を Windows 系計算機に置き換えるという要望が出てきた。

"Data sharing methodology for UNIX and WindowsNT"
Kaoru TSURU, Takehiko HOSOKAWA
Mitsubishi Electric Corp. IT R&D Center

3. 検討の前提

2 に示したようなシステムにおいて、同一アーキテクチャーで構成されたシステムの一部を異なるアーキテクチャーの計算機と置き換える場合には、異機種間データ変換を考慮してプログラムの書換えを行う必要がある。

今回検討を行った対象システムに関して、プログラムの書換えが必要となる箇所としては、以下がある。

(a) ネットワーク対応の独自ファイルシステム内のレコード入出力部分

(b) ネットワーク対応の独自プロセス間通信のデータ入出力部分

検討を行う前提としては、適用コストの最小化を図る必要性から、M/Wでのデータ変換を行うという観点で検討を行うこととした。なお、UNIX から WindowsNT へのプログラムの移行に関しては、ソースコードを共通化する観点から別途検討を行った。

4. 問題点

検討対象システムにおいて、UNIX と WindowsNT 間でのデータ共有を実現するに際して問題となるのは、以下のような項目である。

(1) Endian への対応

UNIX RISC 系 CPU では big endian、Intel 系 CPU では little endian でありバイト並びが異なる点。

(2) 64bit データと 32bit データ

64bit 長整数型をサポートしている UNIX 系 OS と 32bit 長整数型のみをサポートしている Windows 系 OS 間でのデータ整合性。

(3) 日本語コード

現在、既存 M/W、App は、日本語 EUC で統一されているが、Windows 系 OS では、日本語 EUC がサポートされていない点。通常は、メッセージカタログシステムなどを使用して、言語依存部分をソースより排除すべきであるが、現状の M/W、

App はそのようになっておらず、日本語がソースコードにハードコーディングされている。

なお、今回は、(1)の項目に絞って検討を行った。

次にM/Wでデータ変換に対応する場合に問題となるのが、ファイル内のレコード構造や通信データの構造は、App が管理しており、M/Wには単なるバイト並びとしか認識できない点である。従って、M/Wが各入出力データに応じてデータ変換を行なうには、その構造を知る手段が必要となる。

5. 解決案

[入出力データとデータ構造の対応]

・システム内での個々のファイルはファイル名のみで識別可能であるため、ファイル名に対応する App の構造体定義と一対一に対応付けを行う。

・通信データに関しては、識別情報が付随する場合とない場合があるため、識別情報がある際には、App の構造体定義と一対一に対応付けが可能となるが、識別情報がない場合には、M/Wのインタフェースを拡張して対応する。

[データコードの統一]

図2に示すように、WindowsNT のメモリ上のデータ(メモリファイル内レコード)のみ、ローカルなコードとし、ディスク上ファイル内レコード、通信データは、big endian 系に統一する。

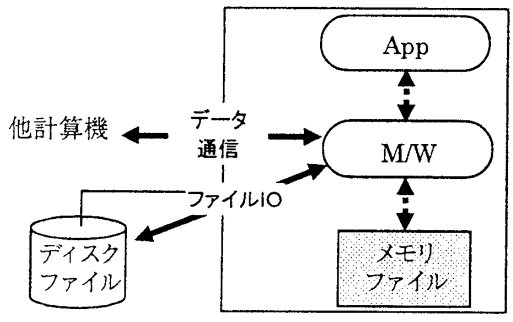

図2

[アプリケーション側の負荷を軽減する方策]

アプリケーション側で行なうのは、データ識別子(ファイル名、通信データ識別情報など)とデータ構造(C言語の構造体定義)を関連付ける"データ構造定義ファイル"を用意するのみとし、M/Wへのコード変換ルーチンの組み込みを行なうコマンドを用意する(図3)。

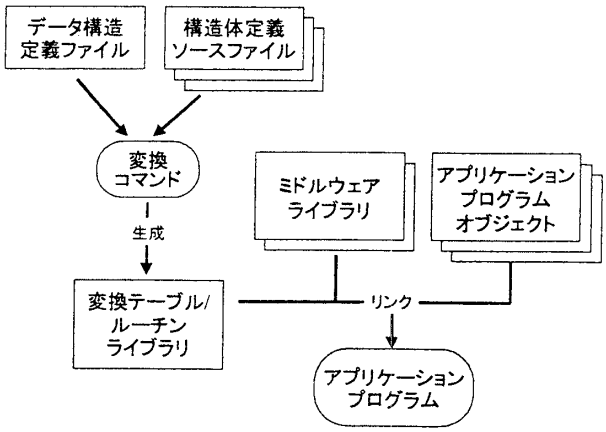

図3

[実装]

図 4 に示すように基本的に入力側と出力側の変換ルーチン群とこれを呼出す変換ルーチンへのエントリー部分("WtoU"と"UtoW")を自動生成する。

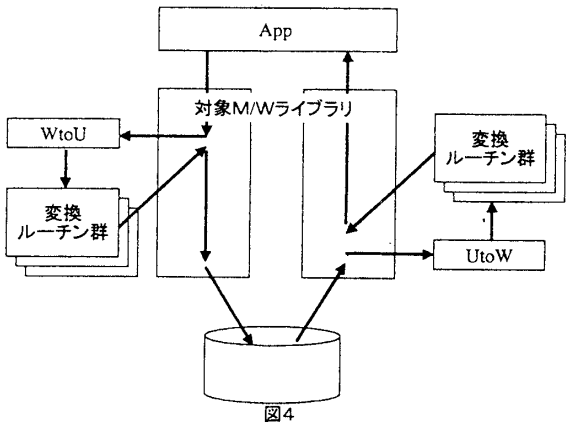

図4

6. まとめ

M/Wにおいて、アプリケーションのデータ変換を簡易に行う仕組みを組込むことにより、システムの一部をWindows 系計算機に置き換える際に発生するデータ共有問題を低コストに解決する手段を提案した。

7. 今後の課題

問題点(2)の 64bit データと 32bit データに関しては、具体的な実装方式に関する検討が更に必要である。また、問題点(3)の日本語コードに関しては、現状で最も簡易な方策として、シフト JIS に統一する方向で考えている。しかし、現在の日本語コード体系に関する論議、動向を見ながら更に検討を進めていきたい。