UNIXとWindowsNT 間のデータ共有方式の実装

1U－3

細川 武彦　鶴 薫
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。

本稿では、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として ミドルウェア(M/W)でデータ変換を行うためのＡＰＩのプロトタイプを開発し、評価した結果について報告する。

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。

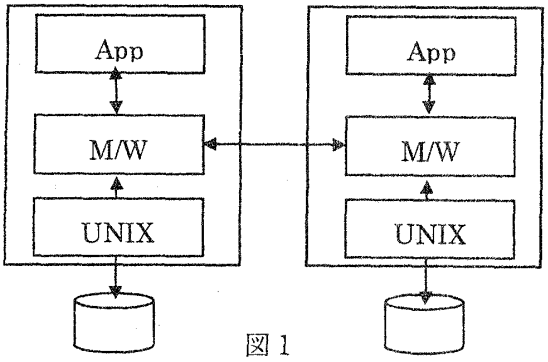

図１

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。

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

3. 実装方式

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で構成されたシステムの一部を異なるアーキテクチャ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き換えを行う必要がある。この時、適用コストの最小化を図る必要性からM/W でデータ変換を行い、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"）を自動生成する方式（参考文献[1]参照）により実装を行った(図２)。

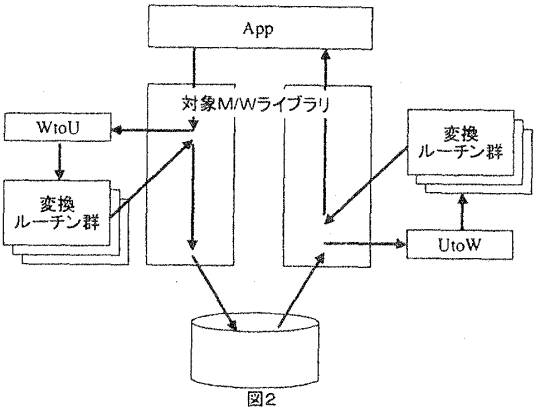

図2

プロトタイプでは、big/little endian、32/64 ビットデータ型の変換に対応し、変換ルーチンを作成する方法、変換の方法として以下のような方式により実装した。

・変換ルーチンの作成方法

A)　コンパイル方式

各定義ファイルから変換用のＣソースコードを作成し、アプリケーションはM/W のライブラリとともにコンパイル・リンクすることにより変換ルーチンを使用する（図3)。

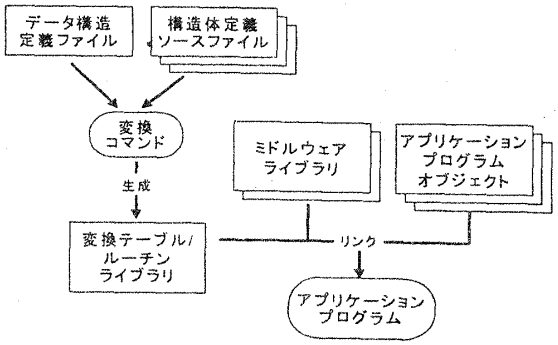

図3

B)　インタプリタ方式

各定義ファイルからバイトコードを作成し、アプリケーションは M/W のライブラリ、バイトコードを実行するライブラリをコンパイル・リンクすることにより変換ルーチンを利用する（図4）。

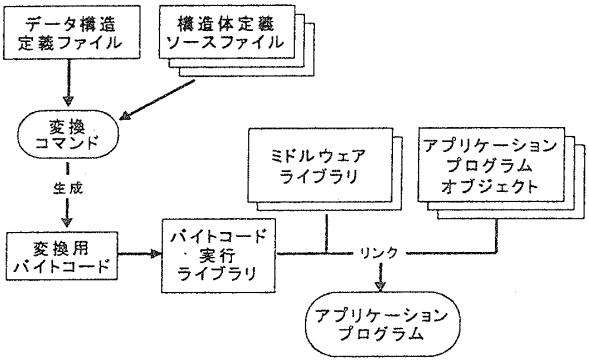

図4

・変換方法

a)　IN／OUT バッファ共有方式

API では、IN/OUT 共有の一つのバッファを指定する。1つのバッファと作業領域を用いて、変換が必要なデータのみ作業領域を用いて変換する。64 ビットデータから 32 ビットデータへの変換では使用できない。

b)　IN／OUT バッファ非共有方式

API では、IN 用、OUT 用の2つのバッファを指定する。2つのバッファを用いて、データのコピー及び変換を行う。

4. 評価結果

3で示したような変換方法を用いて、実際に M/W で使用している約３００の構造体について、big/little endian 変換にかかる時間を計測し、どの変換方法の効率が良いかの評価を行った。

測定環境は以下の通りである。

計算機	HP-9000/K400
CPU	PA-7200 (200MHz)
OS	HP-UX11.0(32 ビット)

測定結果は以下の通りである。

方式	バッファ	変換時間[nsec/byte]			相関係数
		平均	最大	最小	
コンパイラ	共有	140	793	0.002	0.620
	非共有	61	505	6.343	0.836
インタプリタ	共有	550	1983	0.004	0.636
	非共有	379	1197	6.419	0.694

測定の結果より、以下のようなことがわかる。

－　評価した M/W については、コンパイラ方式の IN/OUT バッファ非共有方式が最も効率が良く、このときの性能は 15.6[MB/sec]である。データの変換を行う必要があるのは、ネットワーク通信、共有ディスクへのアクセス時であり、これらの性能が 8~10[MB/sec]であることからデータ変換のオーバヘッドがボトルネックになることはないと考えられる。

－　インタプリタ方式ではコンパイラ方式より 2~6 倍変換に時間がかかる。IN/OUT バッファ非共有方式の場合でも性能は 1.29[MB/sec]であり、データ変換のオーバヘッドがボトルネックになる可能性があるため、インタプリタ方式を使用する場合には性能についての検討が必要である。

－　コンパイラ方式、インタプリタ方式とも、平均変換時間は IN/OUT バッファ非共有方式の方が効率が良いが、最小変換時間は IN/OUT バッファ共有方式の方が効率が良い。このことから構造体内のデータの傾向により、変換方式を選択することで効率を上げることが可能であると考えられる。

5. 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として API のプロトタイプを幾つかの方式により作成し、性能評価を実施した。評価した M/W については、コンパイラ方式の IN/OUT バッファ非共有方式を選択することにより実システムへの適用が可能である。しかし、インタプリタ方式しか選択できないようなアプリケーションの場合には性能についての対策が必要である。

6. 今後の課題

今後、作成したプロトタイプを M/W に組み込み、システムとしてのオーバヘッドについての評価を行う必要がある。また、データの構造や M/W の構造によって、最適なアルゴリズムを選択する方法などについても検討する必要がある。これらの評価・検討を行い、実システムへの適用を進める予定である。

参考文献 [1] 鶴 薫、細川 武彦：「UNIX と WindowsNT 間のデータ共有方式」、情報処理学会第５８回全国大会