解 説

高速エミュレーション技術ー FX!32 ー

The Introduction of Transparent Execution Between Platforms by Virtual Machine by Hirokazu SETO (Software Engineering Group, Digital Equipment Corporation, Japan).

瀬 戸 弘 和[1]

1 日本ディジタルイクイップメント(株)

1. はじめに

異なるプロセッサ上で同一のアプリケーションを透過的に実行する環境は，一般にエミュレータによって実現されてきた．バイナリトランスレーション技術を使って実現した例もあるが，いずれもソフトウェア的な手法で実装されてきたのでオーバヘッドや制限事項が多く，プロセッサが本来もつ性能を十分発揮させることが難しかった．たとえば，エミュレーションではプロセッサ命令を逐次置き換えることによって透過的な実行環境を実現できるが，エミュレータのもつ制限事項や性能に依存する．実行性能を向上させるにはプロセッサ性能を上げる必要もある．一方，バイナリトランスレーションには，実行前に「変換」作業が必要になる．エミュレータのように逐次アプリケーションを実行できる柔軟性はないものの，アプリケーションがバイナリとして静的に蓄積されるため「最適化」を行い，実行性能を向上させる余地がある．このように，エミュレーション／バイナリトランスレーション技術には長所／短所があるものの，既存のソフトウェア資産を有効利用し，プロセッサアーキテクチャ変更にともなう問題を可能なかぎり少なくするために用いられてきた．

しかし，現在のエミュレーション／バイナリ変換技術は，マイクロプロセッサ技術の急速な進歩による劇的な性能向上によって，既存プロセッサ上でのネイティブ性能を凌駕することが可能になり，その目的を大きく変貌させつつある．DIGITAL FX!32 はエミュレータとバイナリ変換を組み合わせた技術だが，高いプロセッサ能力をもってはじめて実用化できた技術でもある．

本稿は，Windows NT/Alpha 上のバイナリ変換技術である DIGITAL FX!32 の技術解説を，Windows NT 上にすでに存在するエミュレータや仮想マシン，今後のプロセッサ動向を含めて解説する．

2. Windows NT の構造と RISC プラットフォームの問題点

Windows NT は発表当初から高い移植性とともにマルチプラットフォームに対応していることを特徴にしてきた．Windows NT 以前には，UNIX がマルチプラットフォーム上で利用されてきたが，UNIX 同様の高い移植性を保ちながら，同一の API とユーザインタフェース(ウィンドウシステム)を異機種間で利用できるのが，Windows NT の大きな特徴である．最新バージョンの 4.0 では，x86/Alpha/MIPS/PowerPC のプロセッサアーキテクチャをサポートしている．(今後のバージョンアップは x86/Alpha を中心に行われる．)

Windows NT は，本格的なフル 32 ビットオペレーティングシステムとして Win32 サブシステムを中心に構成されている．そのほかのサブシステムは Windows NT の実行上妨げとならないのに対し，Win32 サブシステムはそれ自身がスタートしないと Windows NT 自体が利用できない構造となっている．

さらに，Windows NT には x86 系の 16 ビット資産を継承するためのエミュレーション環境として，WOW(Windows on Win32)が Win32 サブシステム上に実装されている．ただし，マルチプラットフォームに対応した Windows NT とはいっても，Alpha プロセッサ上では x86 プロセッサの 16 ビットコードをネイティブに実行でき

ないので，16ビットアプリケーションの実行性能が低かったり，機能上いくつかの制限事項がある．

さて，Windows 95やWindows NT 4.0の出荷により，x86アプリケーションは本格的な32ビット時代を迎えることとなった．1997年10月時点でマイクロソフト社製アプリケーションはほとんど32ビット化(Win32 API対応)を完了している．Windows NT/Alphaでも，ソースレベルでの互換性は確保されているが，x86プラットフォーム上で利用されているWin32アプリケーションに，必ずしもAlphaバージョンが存在しているわけではない．x86プラットフォーム用のWin32アプリケーションはバイナリが異なるためAlpha上で実行することはできなかった．FX!32は，x86プラットフォーム用のWin32アプリケーションをWindows NT/Alpha上で実行する環境を提供するので，ユーザはバイナリの違いを意識せずにアプリケーションを利用できる．このように，FX!32はWindows NT/AlphaにおけるWin32アプリケーションの透過性を高める技術である．

3. FX!32の仕組み

FX!32は，主要構成要素としてFX!32エージェント，FX!32サーバ，ランタイムエミュレータ，バックグラウンドオプティマイザなどがある．FX!32のアーキテクチャは，ランタイムエミュレータでWin32 APIに対応したx86アプリケーションを透過的に実行し，静的に蓄積された変換イメージをバックグラウンドオプティマイザで最適化することによって，次回実行時のアプリケーション性能を向上させるというものである．FX!32は言語に依存しない設計になっているので，日本語Windows NTが前提となるアプリケーションも実行できる．しかし，デバイスドライバやデバッガ，スクリーンセーバ，多言語入力インタフェース(FEP)などはサポートしていない．

FX!32を使用したアプリケーションの実行から，最適化，2回目以降の実行を段階的に解説していくことにする．

3.1 アプリケーションの実行(初回)

Win32 APIに対応したx86アプリケーション(以下，Win32 x86アプリケーションと略す)が起動されると，FX!32エージェントによってランタイムエミュレータが起動される．ランタイムエミュレータは，Win32 x86アプリケーションの実行ファイルと，関連するDLL(Dynamic Link Library)をロードする(これらを「イメージ」と称する．)．同時に最適化コードの有無を検索し，データベースに最適化コードがなければ，エミュレータでWin32 x86アプリケーションを実行する．アプリケーションがエミュレータで実行されている間，使用された関数などをアプリケーションのプロファイル情報として収集しデータベースへ保存する．

3.2 最適化

アプリケーションを正常終了させると，FX!32サーバがデータベース内の新しいプロファイルをチェックする．プロファイルが新規の場合には，オプティマイザの実行が準備される．オプティマイザは，置き換えられたバイナリをバックグランドで自動的に最適化するので，ユーザが特別な操作や設定をしなくても最適化されたイメージを手に入れることができる．オプティマイズ(最適化)によって生成されるファイルはバイナリデータで，Alphaネイティブの DLLとしてデータベースに保存されレジストリ(機器設定やソフトウェア設定が格納された一種のデータベース)に記録される．

FX!32はアプリケーションの実行ファイル自体を書き換えるわけではないので，最適化されたイメージだけで実行することはできない．なぜなら，FX!32はWin32 x86アプリケーションの設定をそのままWindows NT/Alpha上のレジストリに展開し，利用するからである．よって，FX!32を利用するためにはシステム上にWin32 x86アプリケーションが常にインストールされている必要がある．このため，FX!32はWin32 x86アプリケーションのインストールをx86プラットフォーム同様に行う機能も提供している．

最適化作業によってバイナリの最適化が進み，実行スピードをあげることができる．最適化によって最もよい性能を発揮するのは，アプリケーションの性質にもよるが，2，3回程度最適化を終えた後である．これは，エミュレーション時に収集されるプロファイルが，最適化作業の量と範囲を決定するためである．つまり，使われなかった

アプリケーション機能のプロファイルは収集されないため最適化も行われないが，頻繁に使われる機能についてはプロファイルがそのつど更新されるので最適化されやすくなる．

また，最適化作業自体はイメージ全体をすべて最適化するので，CPUとメモリを大量に消費する．このため，付属のスケジューラで最適化作業の開始を業務時間外に設定しておけば，システム性能の悪化が防止できる．

システムがネットワーク環境で運用されている場合には，データベースパスをネットワーク上の別マシンに設定することができる．データベースパスの設定は複数可能で，パスがリモートコンピュータに設定されている場合，FX!32サーバが自動的に最新の最適化コードを選択し，よりよい条件で2回目以降の実行を行うことができる．

FX!32マネージャを使って最適化コードをエクスポート／インポートすることができるので，あらかじめ最適化されたコードを複数のスタンドアロンシステムに配布することも可能である．

最適化コードを共有する機能は，CPU性能やメモリ容量などのシステム資源に余裕がないシステムで有効である．

3.3 アプリケーション実行(2回目以降)

Win32 x86アプリケーションが起動されると，FX!32エージェントによってランタイムエミュレータが起動される．ランタイムエミュレータはWin32 x86アプリケーションの実行イメージと関連するDLLをロードする．同時に，最適化コードの有無を検索し，データベースに最適化コードがみつかると，データベースに蓄積されているAlphaネイティブコード(DLL)をロードする．最適化コードがない場合，あるいは利用できない状態のときにはエミュレータが起動される．

2回目以降の最適化には直前のプロファイルが使用され，最適化されたバイナリでデータベースが上書きされる．最適化を重ねると，アプリケーションの実行性能も向上するが，それにともなってプロファイルやバイナリのサイズも大きくなるため，ディスク容量には注意が必要である(FX!32マネージャの機能でディスク使用量の上限を設定することができる．)．

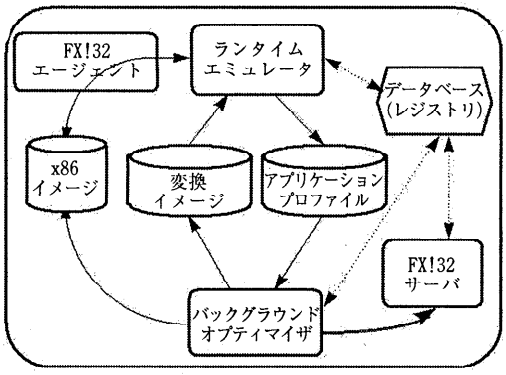


図-1

4. 各部の概要

FX!32の構成要素の関係はこのようになる．各構成要素をそれぞれ説明していくことにする．

4.1 主要構成要素

FX!32 エージェント

FX!32エージェントは，Windows NT/Alpha上のすべてのプロセスに常駐し，Win32の「CreateProcess」コールを検知する．x86コードが検出された場合，ランタイムエミュレータが起動される．

FX!32 サーバ

FX!32サーバは，Windows NTのネイティブなサービスとして設定され，アプリケーションプロファイルの収集や最適化作業のスケジューリングを行う．

ランタイムエミュレータ

ランタイムエミュレータは，エミュレーションだけでなく変換されたイメージとのインタフェースを受けもつ．x86コードをロードし，ロードされたx86コードをAlphaネイティブに置き換えるインタプリタエンジンと，変換イメージがある場合はそれをロードする機能を提供する．Alpha上で利用可能なシステムサービスやライブラリとのインタフェースも受けもつ(これをジャケットと呼ぶ．)．

バックグラウンドオプティマイザ

FX!32サーバが収集したプロファイルデータをもとに，x86コードをAlphaネイティブコードに置き換える．置き換えられたコードはAlphaネイティブのDLLとして扱われる．

データベース

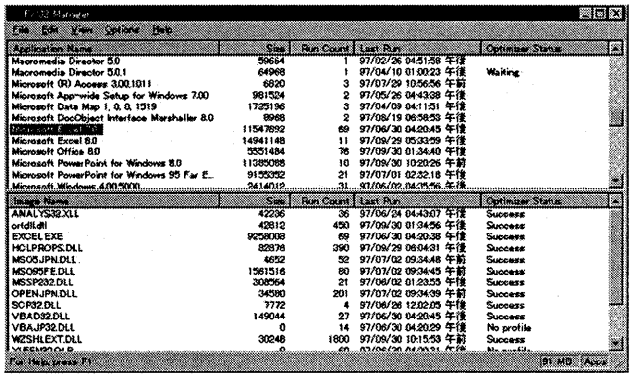


図-2 FX!32 マネージャ

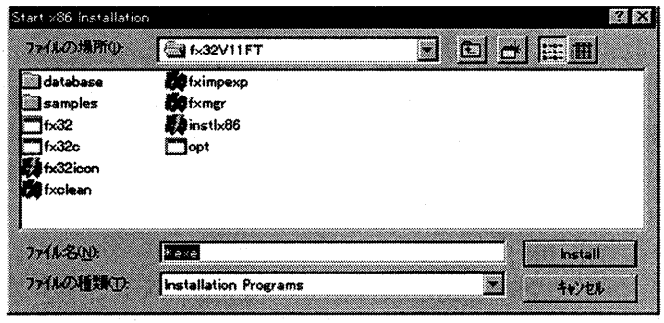


図-3 Install x86 Application

収集したプロファイルおよび置き換えたコードは，Windows NT のレジストリに書き込まれ，次回の実行時には保存されたコードが使用される．

4.2 そのほかの構成要素

FX!32 には，FX!32 の状態を一覧するアプリケーション(FX!32 マネージャ)と，x86 アプリケーションのインストールを行うユーティリティ(Install x86 アプリケーション)が用意されている．

FX!32 マネージャ

FX!32 マネージャは，FX!32 に取り込まれている Win32 x86 アプリケーションのプロファイルや最適化コードなどの情報を GUI で管理するアプリケーションである．最適化作業のスケジューリングや，コードの削除／エクスポート／インポートのほか，ディスク容量の管理，最適化時間の設定などが行える．オンラインヘルプおよびメニューは現在のところ英語表示のみである．

Install x86 アプリケーション

Win32 x86 アプリケーションのインストレーションプログラムを起動するユーティリティである．FX!32 を利用するためには，Win32 x86 アプリケーションが正常にインストールされている必要がある．Win32 x86 アプリケーションのインストールを正常に行うために，FX!32 は Win32 の「GetSystemDirectory」というコールによって得られるパスを，%SystemRoot%¥Sys32x86 というパスに変更する．この操作によって，Alpha ネイティブのシステムディレクトリに x86 ファイルが混入するのを防いでいる．とくにシステムの実行に大きく影響するシステムファイルは，Windows NT の場合異なるプロセッサ用でもファイル名が同一なので，このような「交通整理」が必要になる．

つまり，x86 システムでインストレーションを行うと，%SystemRoot%¥System32 にコピーされるファイルは，FX!32 の場合，%SystemRoot%¥Sys32x86 へコピーされる．Sys32x86 ディレクトリにはいるのは，あくまでも「GetSystem Directory」によってインストール先を判断するモジュールのみなので，Win32 アプリケーションのインストール先は，この関数によって影響を受けない．

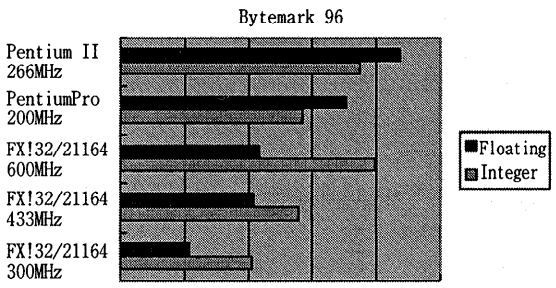


図-4

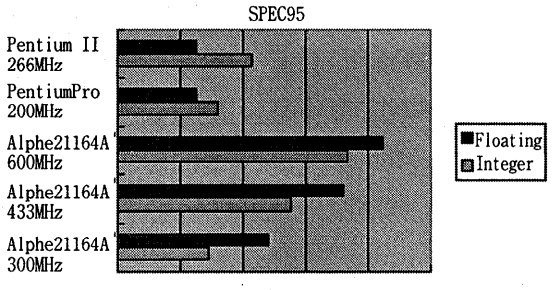


図-5

しかし，一部のWin32 x86アプリケーションでは，システムディレクトリの判別にこのルーチンを使わず，インストール先のディレクトリがアプリケーションプログラムにハードコードされているものがあるため，「Install x86 Application」コマンドを使っても正常なインストールが行えないものがある．

5. 実行性能

FX!32上でWin32 x86アプリケーションを実行した場合の実行性能や最適化に要する時間は，バイナリファイルの大きさとCPUの能力に大きく依存する．最新のAlphaチップは21164 600MHzであるが，FX!32を快適に利用するためには，21164 300MHz以上のチップと64MB以上のメモリを推奨する．

以下は，それぞれFX!32の実行性能(米Byte誌のBytemark 96で計測)と，プロセッサの絶対性能(SPEC95による性能指標)である．Bytemark 96は一般によく知られている米Byte誌のベンチマークで，ベンチマークプログラムも公開されている(http://www.byte.com/bmark/bmark.htm)．数値は実際にFX!32上でx86用のファイルで計測したものである．SPEC95はプロセッサの絶対性能を比較する指標で，ベンチマークプログラムや主要なプロセッサの計測結果が公表されている(http://www.specbench.org/spec)．いずれも，数値が大きいほうが性能がよいことを意味する．21164A 500MHz上で最適化コードを用いて計測した数値は，PentiumPro，Pentium IIのネイティブ性能に匹敵している．また，異なるクロックスピードをもつ21164プロセッサ同士では，FX!32の性能格差は絶対性能値の格差を反映していることがわかる．

6. 仮想マシンの現状

FX!32以外にも仮想マシンを提供する技術が脚光を浴びている．第1に，SUN Microsystems社がライセンスするJAVA VM(Virtual Machine)がある(http://java.sun.com)．JAVA VMは，JAVAでプログラミングされたコードを，プラットフォーム／オペレーティングシステムごとに用意されるVM上で実行するアーキテクチャをとる．たとえば，Windows NT/Intel上のJAVA VM上で稼動するJAVAコードは，Windows NT/Alpha上のJAVA VMでも実行できるので，同じオペレーティングシステムではプロセッサの違いによる性能向上(スケーラビリティ)を期待できる．

ただし現段階では，Windows NT上のJAVAコードは，JAVA VMがもつインタプリタによって実行されるので，高い性能を期待することはできない(Windows NT上のJAVA VMは，Internet Explorer 3.02の追加オプションか，Internet Explorer 4.0で提供されている．)．性能と最適化の問題は，プロセッサアーキテクチャとオペレーティングシステムごとに用意されるJIT(Just In Time)コンパイラに大きく依存する．JITコンパイラでは，JAVAコードをネイティブコードに置き換えるので，プロセッサの能力を生かした実行性能が得られる．SUN Microsystems社では，JITコンパイラ以外にもJAVAコードをネイティブに実行するJAVAチップを提案しており，組込み型のMPUとしての用途に活路を見い出そうとしている．

一方，米インテル社では初の64ビットプロセッサを，ヒューレットパッカード社(以下HP)と共同で開発している．既存のIA-32に対してIA-64と呼ばれるアーキテクチャをもとに，1998年

中には開発を終えるといわれているプロセッサ(コードネーム:Merced)である．Merced に関しては，執筆時点では詳細が明らかになっていない(1997 年秋のマイクロプロセッサフォーラムで Merced の詳細が発表される模様．)．現在判明しているのは，VLIW(Very Long Instruction Word)アーキテクチャをとり，既存の 32 ビットアーキテクチャとは仮想マシン(バイナリ変換)で整合性をとるというものである．

VLIW は既存の x86 あるいは PA-RISC とは異なる命令セットなので，x86 および PA-RISC 用に開発されたアプリケーションとの互換性を保持できない．このため，既存ソフトウェア資産の継承方法としてなんらかの仮想マシンが実装される模様だ．詳細は不明だが，x86 あるは PA-RISC コードを実行する変換ユニットをチップ上に実装されるといわれている．このような手法が可能になったのは，集積技術の進歩とプロセッサ性能の飛躍的な向上により，仮想マシンという元来複雑でオーバヘッドの大きい手法をとっても，既存の 32 ビットプロセッサの能力を発揮する目処がたったためであろう．

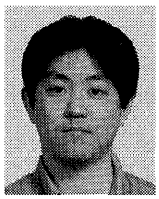
7. FX!32 の今後の予定

さて FX!32 と Alpha チップの今後だが，FX!32 については浮動小数点演算性能の向上が課題としてあげられる．1997 年秋頃に予定されている V1.2 リリースでは，浮動小数点演算専用のエミュレータが導入される見通しである．V1.2 以降については，V2.0 で Windows NT 5.0 の構成要素として統合される予定である．

8. FX!32 入手方法

http://www.dec-j.co.jp/ic/nt/fx32/より，無償ダウンロード可能．

(平成 9 年 9 月 3 日受付)



瀬戸 弘和

1966 年生．1990 年日本ディジタルイクイップメント(株)入社．ソフトウェア製品の販売促進／販売管理，およびワークステーションのマーケティング活動などを行う．現在，ワークステーション製品の製品管理および FX!32 の技術サポートを担当．