4M-2

EWS4800シリーズ －Xウィンドウのチューニング技法－

本郷喜裕

(日本電気株式会社)

1. まえがき

X Window Systemはクライアント・サーバモデルに基づくネットワークトランスペアレントな環境を提供し、移植性の良さとパブリックドメインという配布形態により標準ウィンドウシステムとしての地位を固めつつある。

このたび、当社のUNIXワークステーション(EWS4800シリーズ)においてもX Window Sysyemの移植を行ったのでその移植技術(デバイス依存部のチューニング技術を中心に)について本稿にて報告する。

2. 移植の概要

X Window Systemの移植はウィンドウサーバ(Xサーバ)を動作させることが大部分を占める。

Xサーバの構造は図1のようになっており大きく分けるとdix(デバイス独立部)、ddx(デバイス依存部)、os(OS依存部)となる。移植作業はddxとosを各マシン用に変更することである。ddxはさらにmi(デバイス独立描画系)、snf(フォント処理)、mfb(モノクロ描画系)、cfb(カラー描画系)、nec(入力処理、その他)に分けられている。このように各部分が明確に区別されていることが移植を容易にしている。

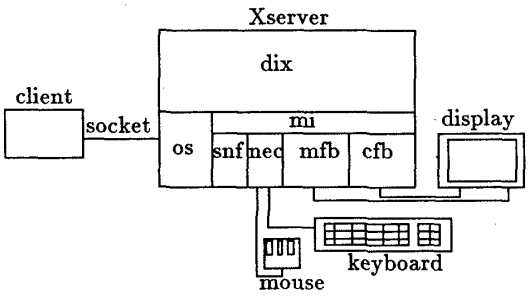

図1 Xサーバの構造

一般に移植作業は次の2段階に分けられる。

・まずは、何とか動作させる。

・実用に耐えるようにチューニングする。

Xサーバの場合、"何とか動作"という段階は以下のような環境が揃っていれば比較的簡単に到達することができる。

(a)プロセス間通信ができる

(b)マウス、キーボードからの入力ができる

(c)フレームバッファのユーザプロセスへのマッピングができる

この3つの条件が満たされていればnec(入力処理、その他)の部分を少しコーディングするだけで動くようになる。(a)については、EWS4800はSystemV系のUNIXをOSとして採用しておりネットワーク機能が不十分であった。これはBSD系のUNIXの標準プロトコルであるTCP/IPを実装しソケットインタフェースを実現することにより解決した。(b),(c)についてはXウィンドウ用のドライバを新たに作成することにより解決した。

一応動作すると次は実用で耐える程度にチューニングを行わなければならない。おもなチューニング部分は描画系である。以下で入力系と描画系におけるチューニングを考慮した実装の方式について述べる。

移植に際しては配布されるドキュメントの移植ガイド[1]やサーバの構造とインタフェースの説明[2]を参考にした。

3. 入力系について

非同期入力としてXサーバが必ず扱わなければならないものは、マウスとキーボードからの入力である。おもな実装方式は次の2とおりである。

(1) システムコールを使う(selectで待ち、readで読む)

(2) ドライバとXサーバで共有メモリを持つ

(1)はマウス、キーボードからの入力を他のクライアントからのプロトコル要求と同じselectで待つことができるが、入力の有無のチェック、読み取りにシステムコールを用いなければならないのでマウスの反応速度が落ちる可能性がある。

(2)はマウス、キーボードからの入力の有無をイベントキューのheadとtailのポインタを比較するだけで知ることができ、入力データも共有メモリを参照することにより得られるのでシステムコールによるオーバーヘッドは生じない。しかし、カーネルのメモリ空間(イベントキュー)をユーザプロセス(Xサーバ)にマッピングする必要がある。

EWS4800での実装方式は以下で述べるような理由によりXウィンドウ用のドライバを作成し、共有メモリによって行っている。

本来、Xサーバのdix(デバイス独立部)は共有メモリによるハードウェアのサポートが可能なように、2つのポインタ(headとtail)を比較して等しくない場合にイベント処理が呼ばれるようになっている。マウスやキーボードの反応速度を落とさないためにも共有メモリを用いるのが好ましい。ただし、イベントがない場合にはXサーバは何か(イベント、クライアントからの要求)が起こるまでselectで待つため、マウス、キーボードデバイスに対するselectコールが行えることが必要である。

EWS4800 Series
- Tuning Technique of X Window System -
Yoshihiro HONGO
NEC Corporation

4. 描画系について

描画系の実装方式について、どのプロセッサがフレームバッファをアクセスするか、またアクセスのときカーネルを経由するかによって図2のような4つの方式が考えられる。

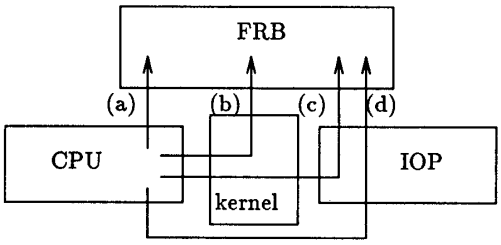

図2　描画の実装方式

(a)CPUがフレームバッファをマッピングし直接アクセスする。

(b)CPUがカーネル内でフレームバッファをマッピングし、描画はシステムコールによってカーネル経由で行う。

(c)IOPがフレームバッファをマッピングし、描画はシステムコールによってカーネル経由で IOPにコマンドを送って行う。

(d)IOPへ送るコマンドを設定する領域がユーザプロセスにマッピングされていて、描画はIOPに直接コマンドを送って行う。

これらの方式は性能、ポータビリティ、コストという観点から考えるとそれぞれ利点、欠点があり目的によって実装方式は異なる。

性能という点では(c),(d)はCPUとIOPの並列動作により高速実行が可能である。また、システムコール（カーネル経由）を用いるよりは直接アクセスのほうが高速である。実際、直線を1本描画するために1回のシステムコールを発行していては性能が出ない。また、大量のデータコピーが必要な場合は問題がある。

ポータビリティという点ではカーネルを経由する(b),(c)はOSのバージョンアップで新ハードウェア対応やチューニングが可能である。また、描画の排他制御がカーネル（またはIOP）で集中管理できる。ただし(b)はカーネルが肥大化するという欠点がある。

コストという点ではIOPを実装している高価なマシンでは性能アップのために(c),(d)の方式を用いることができるが、 IOPを実装していないものは低コスト化が可能なかわりに (c),(d)の方式で性能アップが計れない。

Xサーバのサンプルコーディング（mfb,cfb）は方式(a)であり、フレームバッファをユーザプロセスにマッピングさえできれば動作するように非常にポータブルに作られている。しかし、mfbはそのままでもある程度の性能が得られるが、そのプレーン枚数倍（EWS4800は8プレーン）の処理を必要とするcfbを実際に動かしてみると、かなり遅く実用に耐えられるものではない。

EWS4800はIOP、ラスタオペレータハードウェア、専用アクセラレータ（オプション）を備えたマシンである。今回の移植では性能を重視し(a),(c)の両方の方式を用い、チューニングは次のような方針で行った。

(1)開発期間短縮のため基本的にはmfb,cfbを流用する。

(2)ウィンドウへの描画はラスタオペレータハードウェアを用いる。それによって、1プレーンに描画するのと同じ速度で8プレーンに描画することができる。

(3)CPUとIOPとの並列効果によるスピードアップを狙う。フレームバッファはCPU,IOPの両方にマッピング可能で、しかもXY形式、Z形式のどちらの形式によってもアクセスできる。

(4)より高速なアルゴリズムを利用する（明らかに書き換えることでさらに高速に実行できるものがある。）。

(5)特殊な描画条件（例えば、タイルの幅が16、一点鎖線など）ではハードウェアの効率的利用、アルゴリズムの特殊化が可能な場合があるのでそれを行う。また、dixからddxの描画関数の呼び出しは描画関数のポインタを登録し、この登録されている関数が呼び出されるようになったおり、このポインタは描画条件により容易に切り替えることができる。

チューニングによりカラーシステムの場合、サンブルサーバに比べてかなりの高速実行が可能になった。しかし、現在でも次のような問題点が存在し今後の課題となっている。

- IOPへのコマンドの発行がカーネル経由でしか行うことができない。よって描画に多くの時間を要する処理（システムコールのオーバヘッドが無視できるぐらい）のみしかIOPで実行できない。これは、(d)の経路での実行を検討中である。
- CPUのほうがIOPより処理能力が大きい場合、並列効果があまり期待できない。
- アクセラレータの機能はXでは現在利用していないが、さらに高速化に向けて利用を検討中である。しかし、IOP、アクセラレータで提供される機能のうち利用できるものが限られいるという問題もある（直線、矩形コピー、矩形塗り潰しぐらいしか利用できない）。

5. むすび

X Window Systemは汎用ワークステーション上に移植が容易なことが広く普及した要因の一つである。しかし、真に性能を追及するためには標準のアーキテクチャや描画機能では十分ではなく、Xを意識したものを設計する必要がある。

参考文献

[1] S.Angebranndt,R.Drewry,P.Karlton,T.Newman and B.Scheifler,"Strategies for Porting the X v11 Sample Server",MIT,Mar.1988.

[2] S.Angebranndt,R.Drewry,P.Karlton,T.Newman and B.Scheifler,"Definition of the Porting Layer for the X v11 Sample Server",MIT,Mar.1988.