2V－4

64bit単一仮想記憶OSにおける X Window System のクライアント/サーバー通信機構

友田一郎　津田悦幸　岡本利夫
（株）東芝　研究開発センター　情報・通信システム研究所

1. はじめに

我々は64bitマイクロプロセッサでの利用を想定して、その広大な仮想記憶空間を有効に利用する仮想記憶システムを備えたオペレーティングシステム(Cubix: CUBe of 2 byte unIX [1],[2]) の開発を進めている。Cubixは単一仮想記憶によって、共有メモリとプログラム間にまたがるサブルーチンコールを可能とし、効率の良いIPCの機構を提供している。

我々は、その効率の良いIPCを利用したアプリケーションの１つとして、X Window System の移植を試みている。本発表では、Cubix上のXのクライアント/サーバ通信機構について、特に一般のUNIXなどの場合との比較を中心に述べる。

2. UNIXにおけるXのクライアント/サーバ通信機構

まずはじめに一般的なUNIXにおけるXのクライアント/サーバ通信機構（図1）を簡単に見ておく。

Xのアプリケーションプログラムは多くの場合Xlibを使ってサーバと通信する。Xlibは内部にバッファ（デフォルトで4KB）を持っており、Xlibの各種関数はXプロトコルのリクエストをこのバッファに書き込む。このバッファはすぐにはflushされず、いくつかの条件（たとえばバッファが一杯になったときやリプライが必要なリクエストが出されたときなど）にflushされ、socketに書き込まれる。（クライアントとサーバはストリーム型のsocketで接続されている）socketに書き込まれた通信内容は一端カーネル内部のバッファに蓄えられ、サーバーからの読み出しを待つ。

サーバーはselectで複数のクライアントからのリクエストを待っており、クライアントから通信が届くとそれをsocketから request buffer に読み出す。そして、サーバ内部にはリクエスト一種一種に対応して１つのリクエスト処理ルーチンがあり、それが呼び出される。

サーバ内部で処理が終わると、リクエストの種類によっては、クライアント側にリプライが返される。またサーバからクライアントに送られるものとしてはこのほかに、エラーとイベントがある。クライアント側のXlibでは、ソケットから入ってくるデータを、これら３種類に分け、イベントの場合は、Xlib内部にあるイベントキューに入れ、エラーの場合はエラー関数を呼び出し、あるいはリプライの場合にはそれを呼び出したXlib関数に返す。

3. CubixにおけるXのクライアント/サーバ通信機構

3.1. 単一仮想記憶

UNIXではプロセスがそれぞれ別々の論理アドレス空間を持ち、その間のIPCをsocketという機構を使って行っていた。これに対しCubixでは、64bitの広大なアドレス空間を前提に、システム上の全てのプログラムを単一の仮想記憶空間の中において実行する。こうすることによりCubixでは共有メモリによるIPCが容易に行えるようになっている。また、あるプログラムの中から、別のプログラムのサブルーチンを直接呼び出すことができるようになっている。Xにおいて言えば、Xlibの中からサーバのリ

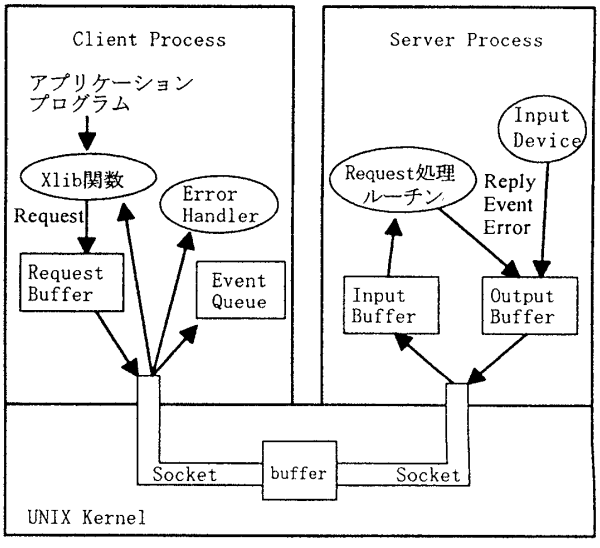

図１. UNIXにおけるXのクライアント/サーバ通信機構

Client/Server Communication Mechanism of the X Window System on A 64bit-Oriented Operating System

Ichiro Tomoda, Yoshiyuki Tsuda, Toshio Okamoto
TOSHIBA Reserch and Development Center
1 Komukai-toshiba-cho,Saiwai-ku, Kawasaki 210, Japan

UNIXはBell研究所が開発し、USLがラインセンスするオペレーティングシステムです。

X Window System は X Consortium, Inc. の商標です

クエスト処理ルーチンを直接呼び出すことができる。リクエストの引数はサブルーチンの引数として、またリプライはサブルーチンのリターンバリューとして受け渡せる。イベントも、サーバプログラムの中からXlib内部のenqueueルーチンを直接呼び出して渡すことができる。UNIXのsocketのようなIPC機構は必要なく、シンプルなクライアント/サーバ通信が可能となる。

6.2. 保護機構

UNIXでは仮想記憶空間を分けることにより、あるプログラムが他のプログラムのメモリ内容を破壊することが防げたが、Cubixでは全てのプログラムを単一の仮想記憶空間に置くため、同一の空間内でも、あるプログラムのメモリ領域を他のプログラムから保護する必要がある。そこでCubixでは、仮想記憶空間内にメモリセクションと呼ばれる領域を作り、このメモリセクション毎に ACL（Access Control List）によりアクセス権の設定ができる。ACLの１エントリは「スレッドID」と「現在スレッドが実行しているプログラムのメモリセクションID」をキーとして「readable」「writable」「executable」の各保護属性が（すなわち、どのスレッドがどのプラムから読み出し/書き込み/実行可能であるかが）設定できるようになっている。なお、このアクセス権チェックは、そのような特別な機能を持ったハードウエア（MMU）により行うことを考えている。

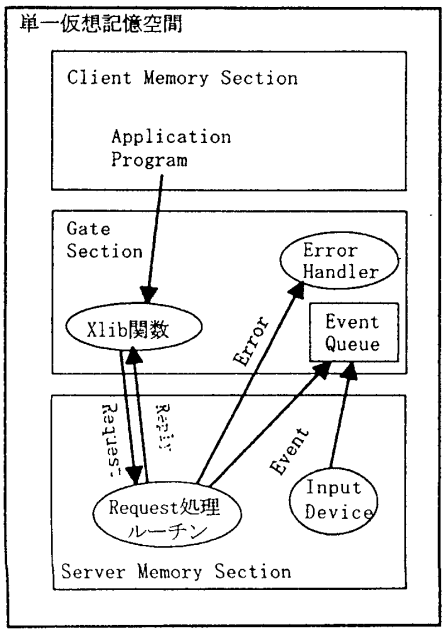

図２. CubixにおけるXのクライアント/サーバー通信機構

われわれのXの実装はおおよそ図２のようになっている。Xサーバ内の全てのルーチンがクライアントから呼び出し可能になっていると危険であるので、クライアントとサーバの間にゲートセクションというメモリセクションを挟み、ここに、Xサーバとクライアントのインターフェイスであるxlibを置く。クライアントからゲートセクションのルーチンを呼び出すこと、および、ゲートからサーバ内部のルーチンを呼び出すこと（及びその逆）はできるようにするが、クライアントからサーバ内のルーチンを呼び出すことはできないようにアクセス権を設定する。こうすることによりXlib以外の方法でサーバ内部にクライアントが入ってくることを防ぐ。

6.3. Cubixの得失

上記のようにCubixでは、UNIXのようにsocketのような通信機構を必要とせず、サブルーチンコールでクライアント/サーバ通信が実現できるため、シンプルで速い。また、クライアントからサーバにまたがるデバッグも１つのデバッガでできるという利点もある。

逆に欠点としては、１つのスレッドがクライアントからサーバ内部に飛んでいくというモデルのため、クライアントの処理とサーバの処理を並列に行うことはできないという点がある。

このようにCubix式のクライアント/サーバ通信メカニズムには得失があるので、これとあわせて、Cubix上にUNIXのsocketと同じものを実装し、これを使ってサーバと通信するXlibも同時に提供する予定である。アプリケーションプログラムはその起動時に、どちらか適切な方のXlibをダイナミックリンクして利用するようにする。

7. 今後の課題

今後はX11R6をベースに、Cubixへの移植を進める。移植に必要な書き換えとしてはクライアントサーバー通信部の他に、フォントファイル/サーバアクセスの部分、色データベースアクセス、キーボード/マウス/スクリーン等のデバイス依存部などがある。

参考文献

[1] Okamoto et al, "A Micro Kernel Architecture for Next Generation Processors", USENIX Micro-kernel and Other Architecture Symposium, 1992

[2]「次世代アーキテクチャ向けオペレーティングシステム・マイクロカーネルの開発」，福本淳他，情報処理学会第47回全国大会 6B-6, 1993