APLワークスペース間通信の一方式

2W−5

金子 聡
日本アイ・ビー・エム株式会社 東京基礎研究所

1 はじめに

APL2は従来、メインフレームのTSSやPC等で使用されたきた、インタープリター言語である。特にTSSではCPUの負荷が高く、レスポンスや他のアプリケーションに与えるパフォーマンス上の影響などの問題があり、一方PCでは、資源の制約がAPL2の使用環境に影響し、必ずしも十分な環境ではなかった。最近ワークステーション上のAPL2言語システムであるAPL2/6000[1]が開発され、これらの問題が解決されつつある。

また、ワークステーションやLANの普及に伴い、分散処理の形態が増加する中で、今後、APL2においてもクライアント-サーバーモデルに基いたアプリケーションの開発が要求されてくると考えられる。

本稿では、そのようなアプリケーション開発を支援する目的で試作された、ソケットインターフェースを提供する補助プロセッサーとこれを用いたAPLのワークスペース間でデータ交換を行なう方法について述べる。

2 APL2/6000

APL2/6000はRS/6000のAIXの下で稼働するAPL2言語システムであり、先に開発されたDOS上で稼働するAPL2/PCをRS/6000上にILと呼ばれる中間言語を用いて移植したものである。X-WindowによるAPL2のセッションを提供し、ユーザーインターフェースの向上が図られている。

3 ワークスペース間通信

従来のホストやPCのAPL言語システムにおいても、分散型のAPLアプリケーションを可能にするため研究が行なわれてきた。例えばESVP[2], GSVPといったAPLで外部とのコミュニケーションを行なう場合の標準として使用される共用変数の拡張に関するもの、あるいは、3270データストリームによるホスト-PC間のデータ交換を行なう補助プロセッサー[3]などである。

ここではIBMのUNIX OSであるAIX上のAPL2をベースとしているため、TCP/IPをコミュニケーションのプロトコルとして採用し、ソケットインターフェースを提供する補助プロセッサー（以下、AP300と呼ぶ）を作成することにより、異なるワークステーション上のAPLアプリケーション間でデータ交換を可能にしている。特に、APL2は対話的なアプリケーション開発が可能であり、これによりクライアント-サーバーモデルに基くAPLアプリケーションを対話的に構築する環境を提供できる。

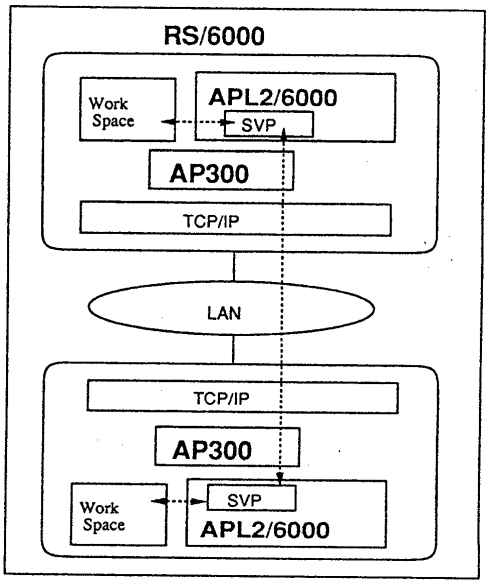

図1: AP300によるワークスペース間接続

図1に示すようにRS/6000がTCP/IPで相互に接続され、双方にAPL2/6000とAP300が導入されるとAPL2/6000のユーザー定義関数は共用変数とAP300を介して、コマンドを送り、データを交換する。

設計に当たっては、次の点を考慮した。

1. プロセス間通信のロジックは、APLの関数で記述できるようにするため、AP300は可能な限り、透過的な処理を行なう。これは、次の理由による。つまり、APLで補助プロセッサーを使用する場合には、通常、カバーファンクションを作成し、ユーザーは、より簡単なインターフェースを使用する習慣があること。また、プログラム開発の観点から、出来る限りアプリケーションに制約を与えないことが必要となることである。

2. コマンドの定義は、APL以外の言語によ

A method for inter-workspace communication in APL
Satoru Kaneko
IBM Research, Tokyo Research Laboratory

るサーバー、クライアントとの接続を考慮して、Cなどで使用しているものに準じた。

表 1: AP300 コマンド

Command	Parameters	Return Value
Socket	domain,soket type host name,port	RC,socket
bind	socket	RC
connect	socket	RC
listen	socket	RC
accept	socket	RC,new socket
send	socket,data,(flag)	RC
recv	socket,flag,(buffer-ptr)	RC,size
close	socket	RC
bufcmd	action, value	RC Rvalue

4 コマンド定義

表 1に AP300 の代表的なコマンドの定義を示す。AP300 との間に共用された変数にコマンドとパラメータを1つのジェネラルアレイとして割り当てる（代入する）ことによって AP300 からソケットのシステムコールが行なわれる。APL のアプリケーション（ユーザー）は通常のソケットシステムコールを使用したプロセス間通信の手順に従って、これらのコマンドを順次、共用変数に割り当てることにより、データ交換のプロセスが実行される。

APL2 側から send コマンドで AP300 に渡されたデータは CDR(Common Data Representation) フォーマットと呼ばれる APL 特有の形式を持ち、receive 側が APL の場合はそのまま CDR を byte 列として、また、APL 以外のケースに対しては APL2 側から文字列の形式でのみデータを渡し、CDR のヘッダーを取り除いて文字列として送出する。これらは flag で制御する。recv コマンドについても同様な制御を行なう。

また、バッファーサイズに関するネゴシエーションは send/recv コマンドの処理の中に組み込んであり、CDR フォーマットで APL アプリケーション間の通信の場合に限って省略可能にしてある。それ以外の場合には、AP300 固有に定義された bufcmd コマンドか APL の関数であらかじめ文字列に変換し、データのサイズを調べた上で、ネゴシエーション行なう必要がある。同様に receive 用のバッファーの獲得、解除も bufcmd コマンドで行なう。

5 プログラム例

図2は異なる WS 上の APL2/6000 のユーザー定義関数による簡単なデータ交換のプログラム例である。ここでは、バッファーの制御を明示的に行ない、HOST1 から HOST2 へデータ（APL オブジェクト）を転送する場合を示している。

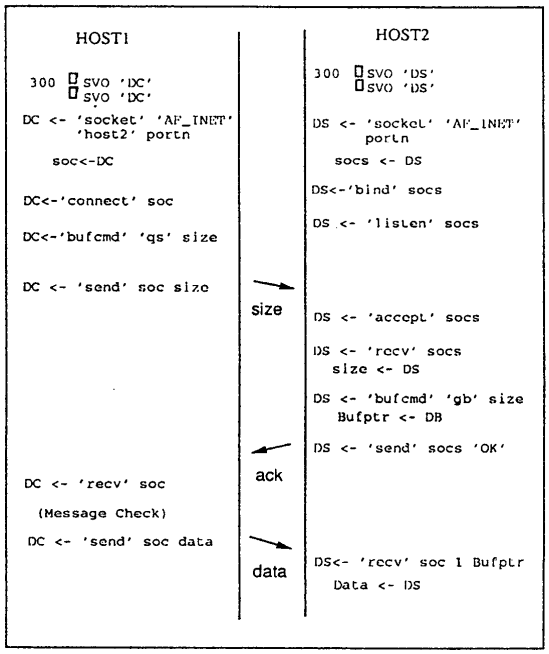

図 2: AP300 によるワークスペース間通信の例

6 おわりに

AP300 を利用することにより、TCP/IP で接続されたワークステーション上の APL2/6000 間でワークスペース間通信が可能となった。特に、対話的にクライアント- サーバーモデルによるアプリケーションの開発ができることが特徴であるが、現状の AP300 では、サーバー側のプロセス生成は APL2 のセッション自体を再起動する方法によるため、この妥当性についてはさらに検討が必要である。また、APL ワークスペースと他のプログラム間の通信に関する改良を加えることなどが今後の課題と考えられる。

7 参考文献

[1] AIX APL2/6000 Users Guide, SC23-3051-0,IBM-Corp.1991

[2] B. J. Hartigan : "Design Considration for an Extended Shared Variable Processor", IBM Research Report, RC13862, 1988

[3] S. Kaneko : "An Experimental Facility for Cooperative Processing in APL" , APL88, 1988

[4] L. E. Zeidner : "The Server Network Generator: A CASE Tools for Distributed Cooperative Processing", APL91, 1991