サービスベースシステムの核の実装とハイレベル化に関する検討

1P-7

深沢　友雄　田中　英彦　元岡　達
（東京大学　工学部）

1. はじめに

共同利用計算機（網）のTSS端末として個人用計算機を利用する時に、それらの間の通信を、各々の計算機の提供する「サービス」の要求／応答という論理的なレベルで統一すれば、ユーザは、両計算機の提供するサービスを、容易に受けることができる。すなわち、ユーザは、個人用計算機を通して、共同利用計算機側のサービスの実現形態に依らず、各々の計算機の提供するサービスを区別することなく利用できる。これを、サービスベースシステムと呼ぶ。［1］

サービスベースシステムでは、基本的に、複数の計算機のコマンド体系が統一したコントロール下にあることが必要となる。この統一したコントロール構造を提供するものを、サービスベースシステムの「核」と呼ぶ。

本研究では、サービスの実体を、

サービス＝アクション＋コントロール＋データ

としてとらえ、アクションを「関数」として扱い、Lisp のコントロール構造を利用した「核」の実装を進めている。本稿では、その実装報告と、それに基づくサービスベースシステムの発展の方向について検討する。

2. システム構成

共同利用計算機、及び個人用計算機のプロトタイプとして、それぞれ東京大学大型計算機センターのM－200H（VOS3）、及びVAX11／780（UNIX）を使用している。基本通信ソフトウェアとして、VAX11側にある、‘CVOS’を利用している。（回線速度9600bps）［2］これにより、M－200H側からは、VAX11がTSSの端末に見えている。又、核の記述言語として、VOS3上のUTILISP、及びUNIX上のFranz　Lisp を用いた。

3. 「核」の実装

3.1 基本方針

双方のLisp のToplevel［（Print（Eval（Read））） ループ］の、（Eval ）を変更し、

・Lisp のFunction として、OSの提供するコマンドを登録／起動できる様にし、

・双方のLisp が、Function Call と、そのReturn Valueのレベルで通信しあうことにより、互いのFunction を、区別なく呼びあえる様にする。

これにより、双方のLisp の Function 、UNIXのコマンド、VOS3のコマンド等を、VAX11側のToplevel のコントロール下に置き、これらを区別なく組み合わせたサービスをユーザが要求できる様にする。（図1．①～⑥）

以下で、これらの実装方法について述べる。

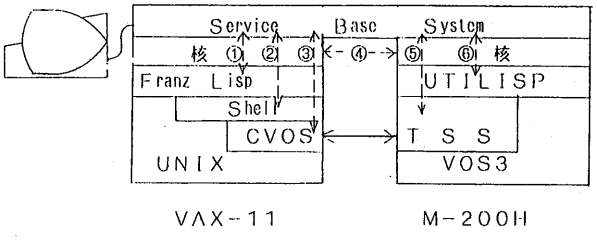

図1　システム構成

3.2 Lisp とOSのインタフェース

各Lisp システムで用意されているOSへのシステムコールを利用することによりOSへのサービス要求を実現できる。このとき、

・引き数として何をわたすか、

・Return Valueとして何を返すか、

が問題となる。

本システムでは、OSが実行する「関数」の引き数として、幾つかのパラメータと、作用を施す対象であるFile 名をわたし、Return Valueとして、OSが作用を施したFile 、あるいは結果をいれてあるFile のFile 名をかえす。この時、標準出力装置への出力データを、本システムが自動的に作成する一時File に入れることにより、Function のNestingを可能にしている。但し、Toplevel の（Print）で、実引き数が一時File 名の時は、そのFile の内容を標準出力装置に出力する様にしている。

3.2 Lisp 間の通信

各計算機の可換性、機能の拡張性を持たせる為に、本レベルの通信は、Simple な形式に統一することが望ましい。そこで、本システムでは、まず相手計算機からメッセージを受ける側のインタフェースと

して、ユーザインタフェースと同様、メッセージを（Read ）し、（Eval ）する方法をとっている。

しかし、一般に、相手からのメッセージは、サービスの「要求」と、自分のサービス要求に対する「応答」との二種類があり、どちらがくるかは、予めわかる訳ではない。（各計算機は、自分がサービス要求を出すと、その応答待ちになるが、処理が、Interpritive に進む為、相手が要求を処理する過程で新たなサービス要求が発生し、「応答」する前にサービスを「要求」してくることがある。）
相手からのメッセージを、単に（Eval （Read ））することによって、「要求」と「応答」を区別する為に、本システムでは、次の様な方法をとった。
すなわち、「応答」する時に、相手が（Eval ）することにより値が得られる様に、
（Quote Return −value ） … （＊）
といった形式で応答する。

しかし、Back End側の計算機のToplevel の（Print）を変更し、いつでも（＊）の形式で応答する方式をとると、Back End側の計算機のTop−level を直接ユーザインタフェースとすることができない。そこで、サービス要求時に、「要求の処理後、結果を（＊）の形式にして（Print）する」といった要求を送ればよい。（図2 II. 2−6）
更に、一般に、相手は「応答」を（Eval （Read ））する状態にいるのだから、要求処理後、再びその状態に戻す様な要求を送ればよい。（図2 I.7）

以上から、サービス要求は、図2のような形式をとっている。

```
(Progn                                          …… 1
  (Print                                        …… 2
    (Cons 'Quote                                …… 3
          (Ncons                                …… 4
            (Apply <func> <arg >) ) )           …… 5
    <outputport>)                               …… 6
  (Eval (Read <inputport >) ) )                 …… 7
```

図2 計算機間におけるサービス要求の形式

3.3 特 徴

・複数の計算機の処理機能が個人用計算機のコントロール下にあり、統一した User 's View を持つ。
・Back End側のToplevel と、Front End側のToplevel が同じ形式で統一されているので、ユーザ側に個人用計算機がなく、普通の端末でも、本システムを使用できる。又、逆に、個人用計算機に、本システムの核を持つ計算機を接続することにより容易にシステム構成を拡張することができる。
・ユーザは、各OSの機能を予め定義された関数として使い、λ -式を用いて、新たな関数を定義することができる。（従来のコマンド プロシージャに相当する。）

4. 実 行 例

```
-> (defs grep ps)
(grep ps)                          UNIXが実行する関数の宣言
-> (defun test (str)               ユーザによる
     (grep str (ps '-s)))          関数の定義
test
-> (test 'lisp)                    関数の実行
  517 a0 S      0:11 lisp

-> (defcent Atom Eq Car Cdr Cons)
(Atom Eq Car Cdr Cons)             M−200Hが実行する
-> (defcent Plus Sub1)             関数の宣言
(Plus Sub1)
-> (defun fib (n)                  ユーザによる
    (conds ((Eq n 0) 1)            関数の定義
           ((Eq n 1) 1)
           (t (Plus
                (fib (Sub1 n))
                (fib (Sub1 (Sub1 n]
fib
-> (fib 5)                         関数の実行
8
```

5. 検 討

核の実装により、サービスにおける、Action のControlの統一が可能となった。 しかし、ユーザインタフェースをハイレベル化する為には、
・各計算機でのデータ構造を共通化する事により、同じ機能を持つAction を対応づける。
・各計算機間で、類似した機能を持つAction を対応づけ、より抽象化したAction として統一する事により、計算機間の差異をシステムが吸収する。
等が必要である。これにより、Data とAction 間のMappingの自由度をあげ、かつ可能なMappingを厳密に記述する事により、知識の利用によるハイレベルなユーザインタフェースを持つ「サービスベースシステム」への発展が期待される。

6. おわりに

本稿では、「サービスベースシステム」において、統一したコントロール構造を提供する「核」の実装について報告し、それに基づく「サービスベース」構成の可能性について検討した。

この検討に基づく「核」の拡張、及びサービスの記述を蓄積していく事により「サービスベース」を構成することが、今後の課題である。

<参 考 文 献>

[1]. 深沢、田中、元岡、「サービスベースシステムの概念と構成法に関する一考察」、昭和56年、10月、第23回情処全大予稿集pp. 613-614.
[2]. 長谷部、石田、岡本、「VAX／UNIXの端末からM200Hを使うためのCVOSコマンド」、昭和56年、9月、東京大学大型計算機センターニュース、Vol. 13 No. 9・10、pp.103-105.