サービスベースシステムの実装

4D-10

荻野　正　深沢　友雄　田中　英彦　元岡　達

(東京大学工学部)

1．はじめに

サービスベースシステム（ＳＢＳ）［１］の有効性を検証する事を目的として、実験システムを実装した。実験システムは、筆者等の研究室のＶＡＸ－１１／７３０（以下７３０）、東大大型計算機センターのＶＡＸ－１１／７８０（以下７８０）、同じくセンターのＭ２８０Ｈの３台から構成されている。この時、核言語としてＬisp を採用した。当システムにより、ユーザは、各計算機上の関数及びＯＳのコマンドを、フロントエンド計算機から区別なく呼び出す事ができる。又、ユーザ環境向上の為、アテンション機構、エラー処理機構等も実装した。

本稿では、これらの実装システムの紹介と、実装方法、ユーザ言語等について報告する。

2．システム構成（図１）

実験システムの７３０及び７８０上には、ＯＳとしてＵＮＩＸ、核言語としてＦranz　Lisp が載っている。又、Ｍ２８０Ｈ上には、ＶＯＳ３、ＵＴＩＬＩＳＰが載っている。

基本通信ソフトウェアは、ＣＯＭ（７３０－７８０間）及びＣＶＯＳ２（７８０－Ｍ２８０Ｈ間）を使用している。ＣＶＯＳ２は、７８０に接続している端末を、Ｍ２８０ＨのＴＳＳ端末として使用できるようにするソフトウェアであり、東大大型計算機センターで開発された。ＣＯＭも同様に、７３０の端末を７８０の端末として使用するソフトウェアであり、当研究室で作成した。

７３０に於いて、Ｆranz　Lisp からＣＯＭを起動する事により、Ｌisp のレベルで７８０への読み書きが可能となる。この時さらに、７８０のＦranz Lisp からＣＶＯＳ２を起動する事により、Ｌisp のレベルで３台が接続される事になる。

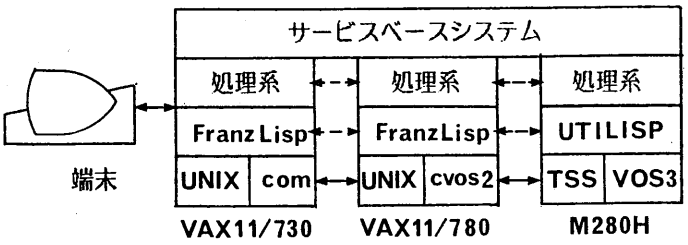

図１　実装システムの構成

3．システム記述言語

実験システムの記述言語は、各計算機のＬisp のＴopＬevelを変更し、

（１）自計算機のＬisp の関数

（２）自計算機のＯＳのコマンド

（３）他計算機の関数、ＯＳのコマンド

を区別なく扱えるようにした。その結果、各計算機の関数、ＯＳのコマンド等を自由に組み合わせたサービスを要求する事ができる。以下、Ｌisp 間の通信について述べる。

サービスの実行中に他計算機の関数が現れたら他計算機にサービスを要求し、その応答を待つ事になる。但し、他計算機へサービス要求を出した時、それに対する相手方のメッセージとしては、その要求に対する応答と新しいサービスの要求とがあり、それらを正しく区別しなければならない。以上の要求を満たす為、次のような要求応答の方法をとった。

まず、他計算機へのサービスの要求は、すべて

（関数名　．引数リスト）

の形式に統一する。サービスの要求の後は次のようなループ（フレーム）を作り相手からのメッセージを待つ。

```
(catch
  (prog  ()
    loop
    (print  (eval  (read  <inport>))
            <outport>)
    (go  loop)))     ……(*)
```

相手からのメッセージは次の形式で返ってくる。

・要求の場合

（関数名　．引数リスト）

・応答の場合

（throw　return-value）

その結果、要求がくれば新たに評価し、（eval）応答ならばフレームから抜けてサービスの実行を続ける事ができる（図２）。

4．システム環境

実験システムで、システム開発環境及びユーザ環境向上の為に実装した機能について説明する。

4.1　エラー処理機構

サービスの起動中のエラー発生に対して

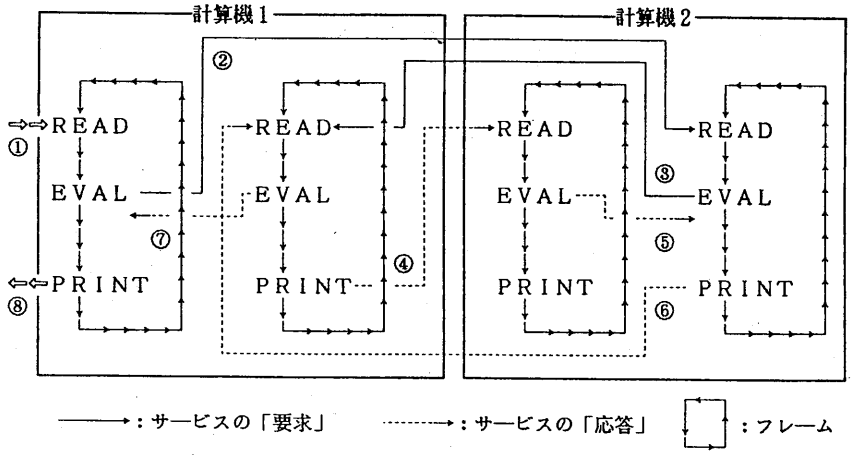

図2　Ｌｉｓｐ間の通信

は、ユーザにエラー発生を知らせて初期状態に戻るようエラー処理機構を実装した。この為には、

①自計算機でエラーが発生した場合に、サービスの要求側に、エラー発生という応答を返す。

②他計算機からエラー発生の応答が帰ってきた場合には、サービスの要求側にエラー発生の応答を返す。

の２つの機能が実現されればよい。①は、各Ｌispのエラー処理ルーチンを変更する事によって実現した。②については、（＊）のフレームを若干変更した。

具体的には、応答を次の２種類に分ける。

・正常な応答

（throw　'return-value　tag ）

・エラー発生時の応答

（throw　'return-value　err-tag ）

エラー発生時のreturn-valueはエラーメッセージ等である。

4.2　アテンション機構

サービスの起動中に、そのサービスを停止させ初期状態に復帰させる為にアテンション機構を実装した。ユーザ端末から発生したアテンションは通信ソフトウェアのレベルで現在通信中の全計算機に伝えられ、各計算機上のＬisp はアテンションを受けるとＴopＬevelに戻るようにした。その後、全計算機がＴopＬevelに戻ったかどうかを確認すればよい。

この機能は各Ｌisp のアテンション処理ルーチンを変更する事によって実現した。

4.3　計算機間のセッションの開始／終了の自動化

実験システムでＳＢＳを起動するには、login（登録番号、パスワード等の投入）やＬisp の起動等の繁雑な操作が必要となる。この操作をあらかじめＦile にいれておく事により、自動的にＳＢＳを立ち上げる関数を実装した。この時７８０とＭ２８０Ｈの間の立ち上げの関数は７８０で定義されており、７３０と７８０の間の立ち上げの関数は７３０で定義されている。ユーザは、７３０を立ち上げる方法と、ＳＢＳを立ち上げる関数の存在を知っていればよい。

５．ユーザ言語

実験システムではユーザがサービスを要求する言語としてＬisp-like な言語を用いている。この言語に於ける制御構造としてcatch とthrow を実装した。

Ｌisp では関数がネスティングしている時の大域脱出の制御機構としてcatch とthrow という関数が実装されている。（throw　tag ）を実行すると（catch　tag ）の所までネスティングをとく事ができる。この制御機構をＳＢＳで実装した。具体的には、まず（＊）のフレーム毎にcatch-tag のスタックを作り、catch が現れたらtag をスタックに積んでいく。throw の実行は、当該フレームのスタック中に対応するtag があれば、そこにthrow し、ない場合には、サービスの要求側にtagまでthrow するよう応答を返す。その結果、ＳＢＳ内のどの計算機にcatch があっても、そこまでネスティングをとく事が可能となる。

６．検討

実装システムに於いてサービスの要求応答の形式はＳimple であり、かつ関数のネスティングに対しても正しく動作する。しかし、エラー発生等の例外的な場合を考えると、この方法が必ずしも最適であるか否かは検討の余地がある。

また、アテンション機構、エラー処理機構等の実装はユーザ環境の向上に役立っている。しかし、現在までのところ、通信ソフトのレベルでの障害対策等については十分に対応できていないのが現状であり、今後の課題である。

７．おわりに

本稿では、ＳＢＳの実装システムについて、その実装方法や言語等について紹介した。現在、より使い易いユーザ環境をめざして改良を進めている。今後は、具体的なアプリケーションの実装を通してシステムの柔軟性や効率等について検討する予定である。

８．参考文献

［１］深沢他「サービスベースシステムの概念と基本構成」、信学技報、ＥＣ８２－４４