4D-8

サービスベースシステムにおける並行処理

深沢　友雄　田中　英彦　元岡　達

(東京大学　工学部)

1. はじめに

我々は以前より、計算機網中に分散して存在するデータやプログラムを任意に組み合わせたサービスを提供できるシステムを「サービスベースシステム(SBS)」と呼び、その構成、及び実現方法等を研究してきた。[1] この時、サービスの実行モデルとして、関数型モデルで取扱ってきた。即ち、或る計算機が他計算機にサービスの要求をした場合、要求された計算機がサービスを実行して、その結果を要求した計算機に返すまで、要求した側は、応答待ち状態となる。この様に、処理が逐次的に進む為、複数の計算機中で同時にサービスを実行させる事ができなかった。このため、パイプライン的処理が実現できなかったり、ファイル転送等の様に両計算機が同時に仕事をする事によって実現するサービスの実行が困難であった。そこで、本稿では、SBSにおいて、複数の計算機上で、同時にサービスを実行させる機構について述べる。

2. SBSにおける並行処理

2.1 並行処理実現の為の必要事項

一般に計算機間の並行処理技術は、計算機網では既知であるが、本研究の様にLispをベースとした言語からの研究は少ない。

並行処理機構の導入にあたって、

①並行にサービスを起動する時の起動のしかた。

②並行動作するサービス間の通信機構の記述と実現。

の2点を考慮する必要があり、次節以下でこれらについて述べる。尚本稿では、複数の計算機上で同時にサービスが実行できるシステムの実現を主眼とし、各計算機内での並行処理に関しては、深く言及しない。後者の実現は、各計算機のOSに依存する。

2.2 並行動作するサービスの起動

SBSでサービスを並行に走らせるには、次のいずれかの方法によるものとする。

①(fork　サービス名)の様に陽に並行に走らせる事を要求する。

②サービスの定義をする時にサービスの属性として、「並行動作させる」という属性を与える。

①に関しては、(cobegin . サービスのリスト)の様な指定のしかたも考えられるが、サービスを起動する時点の明確さ及び、記述能力の点から、①の方法を検討している。並行動作させた場合、あとで(join　サービス名)等により並行動作させたサービスの終了の待ち合わせが必要である。

2.3 サービス間の通信

2.3.1 通信チャネルとポート

サービス間の通信は、下位レイヤが提供する通信径路(チャネル)を用いる。チャネルには、2つの端(ポート)があり、それぞれの計算機でポートを並行動作しているサービスに接続する事により、サービス間が通信経路を確保できる様にする。各サービスは、ポートにアクセスする事により、通信を行なう。

2.3.2 サービスとポートとの接続

或る計算機から他計算機に並行サービスの起動要求を出した場合、要求を受けた側は、起動するサービスが通信を必要とするのなら、その時使うチャネル名を要求元に返す。サービスを要求した側は、返ってきたチャネル名に対応する自計算機側でのポート名を、要求したサービスのreturn valueとする。(図1-①)これを引き数として別のサービスを起動すれば、2つのサービスが並行に動作し、互いに通信する事ができる。(図1-②)　各サービスがポートにアクセスするという事は、サービス定義時に記述しておくものとする。他の方法としては、サービス定義時に両サービスに、アクセスするポートを論理デバイス名として記述し、サービス実行前にそれらを実際のポートに接続する事を陽に記述して、サービスを起動するやり方が考えられる。或いは、各計算機内で並行処理が可能な場合は、あらかじめ、サービスを起動してしまい、それらのreturn valueとしてサービスに対応するポート名を返し、それらをあとで接続していく方法も考えられる。しかし、

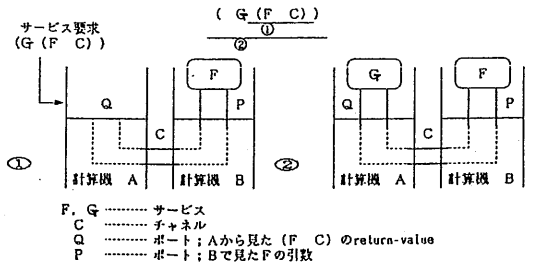


図1　並行動作サービス間の通信

これらの方法は、いずれもサービス間の接続をサービスの起動とは別に記述する必要がある。

本稿で述べた方法は、サービスのreturn valueとして、通信に用いるポート名を返し、次のサービスの引き数としてわたすので、サービスの関数的な性質を用いて記述できる。しかし、各計算機内でも並行動作を認めると、サービス間の接続がより複雑になる事が考えられ、接続情報を別に記述する方がわかり易い場合も生ずるであろう。

3. 並行動作の実装と記述例

3.1 実 装

前章で述べた様な並行動作をサポートする処理系を東大大型計算機センタのＶＡＸ－１１及び、Ｍ２８０からなるシステム上に、Ｌisp とＣ言語を用いて実装中である。この処理系の上では、各サービスの記述はＬisp を拡張した言語によって行なう。サービス間の通信チャネルは、サービスの要求と応答に用いているチャネルを共有している（往復一対のみ）。図１の様なサービス間の通信を行なう場合の処理系間の通信の様子を図２に示す。

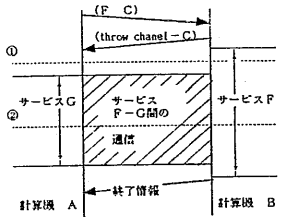


図２ 計算機間の通信

3.2 記述例

サービスの並行起動機能を用いて、

①パイプライン的処理

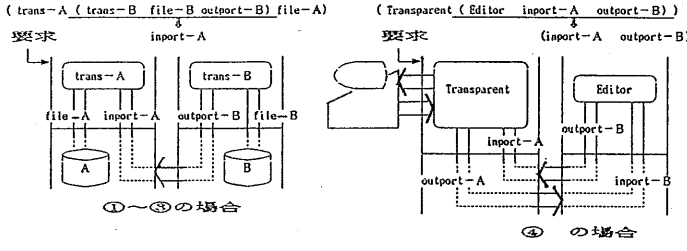
②ファイル転送

③ストリームを転送するサービスの処理

④インタラクティブな処理

が、ＳＢＳの要求言語のレベルでも容易に記述できる（図３）。②～④のサービスは、サービスの定義時にその属性を記述しておけば処理系が自動的に並行にサービスを起動する様にしてある。②～④は、サービスの並行処理機構をサポートしなくても、下位レイヤが処理系に提供している基本通信機能（send とreceive ）を用いる事により、それぞれ別の方法でサポートする事も可能である。[2],[3]しかし、サービスの並行処理機能によりこれらが統一的に扱え、処理系からも他計算機に対して、サービス要求／応答のレベルの通信をする事により容易にこれらのサービスを実現する事ができる。これにより、各計算機は、いつもサービス要求／応答の外部インタフェースだけ用意しておけばよく、今までの様に、副作用を伴なう様な特殊なサービスに対して、特殊なプロトコルを設定する必要はなくなる。

図３ サービスの並行動作例



4. 検討と課題

4.1 ポートと副作用

サービス間の通信は、全てポートを通して行なわれる。この為ファイルや一般の外部機器にアクセスする場合も、アクセス対象をポートとして抽象化すれば、サービスの通信相手がサービスの場合も、その他のデバイスやファイルの場合もまったく同様に扱える。関数型言語という点から考えると、これらは、全て環境に対する「副作用」と考える事ができる。即ち、サービスの並行起動を要求した場合、要求側では、ファイルのオープンと同じ様に見える。この方法により、並行動作するサービス間の通信も、副作用の一種として一般的に取り扱える様になった。

4.2 チヤネルの本数

各計算機内での並行処理を仮定しない為、実装に際して通信径路は、往復一対で十分であった。反面、他計算機で動くサービスを必ず先に起動しなくてはならない。この為、サービスの起動順序と、サービス間のデータの流れが必ずしも一致せず、データの入出力関係が把握しにくい。今後は、各計算機内で並行にサービスが起動される場合について検討していく予定である。この時、複数の通信チャネルの実現方法、更にチャネルの動的生成等についての検討が必要となる。

その他、共有資源へのアクセスの相互排除問題や、デッドロックの回避問題の検討も残されている。

5. おわりに

本稿では、各計算機で同時にサービスを動作させる様な並行処理に関して述べた。この様な簡単な並行処理機能により、より幅の広いサービスの提供が可能になった。又、並行処理により、計算機間にわたるサービスの実行のスピードの向上が期待されるので、速度に関する評価も行う予定である。

＜ 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」、ＩＥＣ８２－４４、pp. 53-62.

[2]．深沢、他、「サービスベースシステムにおける分散データの取り扱いについて」、第２６回情処全大、pp. 811-812.

[3]．大政、他、「サービスベースシステムにおける会話方サービスの実現」、本大会予稿集、４Ｄ－１１.