分散並列OS「Orion」の試作 プロセスサーバの実装

2P−3

山内雅彦，岩嵜正明，吉澤聡，千葉寛之，宇都宮直樹，薗田浩二
(株)日立製作所　中央研究所

1．はじめに

分散環境上で並列処理を行うためには、各ワークステーション（ノード）上にプロセスを割り当て、それらを並列に動作させる必要がある。そのためには、任意のノードにプロセスを生成するリモート・プロセス起動の機能が必要となる。本稿では、リモート・プロセス起動の機能を分散環境上で実現する際の課題とそれらに対する解決方式を示し、試作したプロセスサーバ（PS）のプロセス生成の性能について述べる。

2．リモート・プロセス起動の課題と解決

リモート・プロセス起動には、親プロセスの空間をネットワークを介してコピーする方法と、メッセージによってプロセスの起動を依頼する方法が考えられる。　前者は、Unixのforkシステムコールを分散環境向けに拡張したものである。この方法は、プロセス起動時にネットワークに対して高負荷がかかるという問題がある。また、プロセスのイメージはマシンのアーキテクチャやOSのバージョン等に強く依存するために、システムの移植性が低下するという問題もある。

そこで、Orionシステムでは、メッセージによってプロセスの起動を依頼する方法を採用する。即ち、ユーザによるリモート・プロセス起動要求を、ローカルノード上のプロセスサーバが受け付けると、ユーザが指定したリモートノード上に存在するプロセスサーバにプロセス起動の依頼メッセージのみを伝達する。リモートノード上のプロセスサーバは、そのノード上に新たにプロセスを生成する。

以下に、この方法を採用する際の課題と解決方式の概要を述べる。

(1)環境継承

新たに生成するプロセスのオーナ設定や起動するプログラムファイルのパス名指定を可能にするためには、親プロセスの環境を継承する必要がある。

Orionシステムでは、プロセス生成要求メッセージに親の環境を格納してリモート・ノードのプロセスサーバに伝達する方式を取る。プロセス生成要求メッセージには、親プロセスの環境変数、カレントディレクトリ、ユーザIDを格納する。さらにプロセス生成要求メッセージを可変長とすることで、継承する環境変数の数や長さに制限がないようにしている。

また、SPMD（Single Program Multiple Data stream）のアプリケーションでは、問題領域を分割し、各プロセスはその部分領域を担当する。この時、各プロセスは自分がどの領域を担当するかを判断するために自プロセスの識別子UDN（User Defined Name）を知る必要がある。Orionシステムでは、リモート・プロセス起動時に、生成するプロセスのUDNを環境変数で設定する。

(2)通信ポートの確立

Orionシステムでは、ユーザプロセスが動的にポートを獲得することができるが、ポートを獲得するためのポートを予め割り当てておく必要がある（このポートをデフォルトポートと呼ぶ。また、サーバにはシステム起動時に固定的にデフォルトポートを割り当てている）。

プロセスサーバはプロセス生成時にユーザプロセスのデフォルトポートを以下の手順で確立する（図1参照）。

1）プロセスサーバが新しいポートNを獲得する。

2）プロセスサーバがforkして、サーバの子プロセス（PSレプリカ）の環境変数に、獲得したポートNのポート番号を設定する（即ち、PSレプリカはポートNをプロセスサーバから継承する）。

3）PSレプリカはポートNを保持したまま、execシステムコールを発行してユーザプログラムを起動し、ユーザプロセスになる。

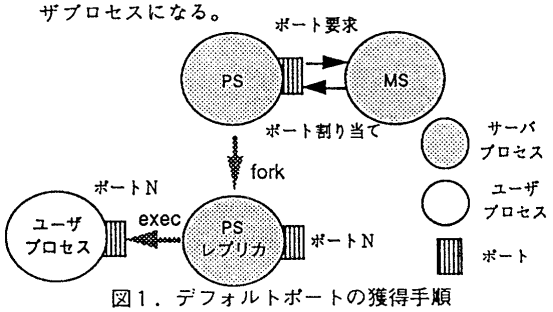

図1．デフォルトポートの獲得手順

1）において、プロセスサーバがメッセージサーバ（MS）からポートを獲得するとき、プロセスサーバがメッセージサーバのポート割り当てを待つブロック方式と、プロセスサーバがポート割り当てを待たずに次の要求を受け付けるノンブロック方式が考えられる。

一般にクライアント／サーバ型のシステムでは、サーバはクライアントからの要求を受け付けると即座に子プロセスを生成し、子プロセスが実際のサービスを行う。これはサービス終了前に、次の要求を受け付けられるようにするためである。

Prototyping of Distributed-Parallel Operating System "Orion"
- Implementation of Process Server -
Masahiko YAMAUCHI, Masaaki IWASAKI, Satoshi YOSHIZAWA, Hiroyuki CHIBA, Naoki UTSUNOMIYA, Kouji SONODA
Central Research Laboratory, Hitachi, Ltd.

プロセスサーバでは、前述のように子プロセス（PSレプリカ）の生成前にポートを獲得する必要がある。そのため、ブロック方式を用いると、ポート獲得の間、次の要求を処理することができない。そこで、ノンブロック方式を採用し、サービスの受け付けとポートの割り当て処理を並行して行う。

(3)シグナル伝達

Orionシステムでは、すべてのプロセスはネットワークワイドな識別子OID（Object IDentifier）を持っている。ユーザは、OIDを利用してネットワーク透過にシグナルを発行する。また、親子プロセス間の同期（exit,waitシステムコール）も、OIDを使用するインタフェースに統一する。

Orionシステムでは、ノード間をまたがるシグナル伝達は、各ノードに存在するプロセスサーバを介在して行う。

また、親子プロセス間の同期は、プロセスサーバが子プロセスの終了をSIGCHLDで検知し、親プロセスを管理するプロセスサーバに通知することによって実現する。

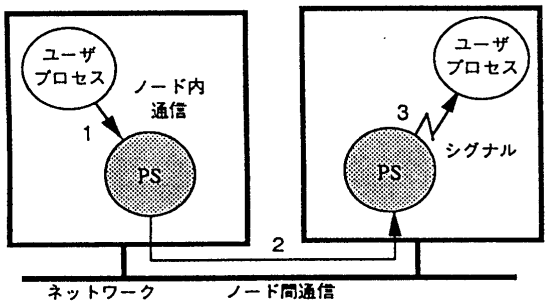

図2．シグナルの伝達

3．プロセスサーバの評価

前述の方式に基づき、プロセスサーバのプロトタイプを実装し、プロセス生成のコストを測定した結果を以下に示す。

ローカルプロセス生成	リモートプロセス生成
2.92 (sec)	3.23 (sec)

ローカルプロセス生成とリモートプロセス生成では、プロセス生成のAPIを位置透過とするため、共にプロセスサーバを経由し、同様のノード内処理を行う。

ここで、ローカルとリモートとで大きなコスト差がないことから、ノード間通信に比べてノード内処理がプロセス生成の大部分を占めていることがわかる。

ローカルプロセス生成のコスト内訳を表1に示す。前述のノンブロック方式によりプロセスサーバは、1と2の処理を行った後は、次の要求を受け付けることが可能なため、連続した要求を迅速に処理することができる。また、デフォルトポートの割り当て処理が全体の９０％以上を占め、隘路となっている。これは、デフォルトポートの割り当て処理において、デフォルトポートをネームサーバに登録する処理が、ネットワークワイドな一貫性制御を伴うためである[2][3]。

プロセスサーバがプロセスをネットワークワイドに管理するためには、ネームサーバへの登録処理は不可欠であり、プロセスサーバの実装においては、デフォルトポートの割り当てコストを低減することが重要である。

表1．プロセス生成コストの内訳

	項目	%
1	UAPプロセスから、PSへプロセス起動要求	1.5
2	MSへデフォルトポート割り当て要求	0.2
3	MSからPSへデフォルトポートを応答	97.9
4	PSがforkしてPSレプリカ生成	0.1
5	プロセステーブルへの登録処理など	0.3
6	UAPプロセスへ生成した子プロセスのOIDを応答	0

＊プロセス生成コストには、execのコストは含まれていない。

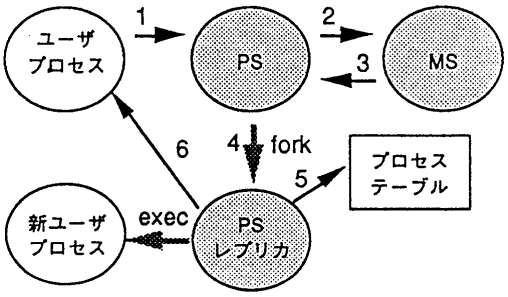

図3．プロセス生成の手順

5．おわりに

本稿では、まずプロセス生成／プログラム起動の機能を分散環境上で実現する際の課題を明らかにし、その解決方式を述べた。

次に、その解決方式に基づくプロセスサーバの実装を行い、プロセス生成のコストを調べた。それによると、プロセス生成の性能は、デフォルトポートの獲得コストに支配されることがわかった。今後、この結果をOrionシステムの設計にフィードバックしていく予定である。

参考文献

[1] 岩嵜, 他, 「分散並列OS「Orion」の試作－システムの概要」, 情報処理学会第45回全国大会予稿集, 2P-01, 1992.

[2] 宇都宮, 他, 「分散並列OS「Orion」の試作－並列計算におけるネームサーバの課題とその解決」, 情報処理学会第45回全国大会予稿集, 2P-02, 1992.

[3] 薗田, 他, 「分散並列OS「Orion」の試作－メッセージサーバの実装」, 情報処理学会第45回全国大会予稿集, 2P-04, 1992.