2P−4

分散並列OS「Orion」の試作 メッセージサーバの実装

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

1．はじめに

OrionシステムではOSに対するサービスの要求やユーザプロセスどうしの通信、さらにプロセス間の同期も全てメッセージ通信を通して行う。即ちメッセージ通信機構は、Orionの核となる機能であり、この性能がシステム全体の性能を左右する。

Orionではメッセージサーバ（MS）が、ネームサーバ（NS）と協調して位置透過な通信、及びユーザ定義名による通信相手の指定等の機能を提供する。このように機能分散型の構成にするとシステムの変更や保守が容易に行なえるといった利点があるが、その反面メッセージサーバのネームサーバに対する問合わせ処理が性能上の隘路になる。我々は、位置情報のキャッシングを行なうことでこの問題の解決を試みている。ここでは、プロトタイプのプロセス間通信機能及びメッセージサーバの構成について述べ、位置情報キャッシングの効果を実測データにより示す。

2．Orionのプロセス間通信機能

Orionではポートを使用した間接通信をサポートしており、次のような特徴を有する。

- 位置透過な通信
- ユーザ定義名によるポートの指定

今回の試作ではUNIXのメッセージキューを使用して上記ポートに相当する機能を実現した。これはノード内で一意な名前を持ち、位置透過性はない。このポートをローカルポートと呼ぶ。

位置透過な通信実現の為には、ポートに対しネットワーク上で一意な名前を付加することと、その名前を用いてポートにネットワーク上のどこからでもアクセス可能である必要がある。Orionでは、このネットワーク一意な名前（OID: Object IDentifier）をネームサーバによって実現し、OIDを用いたポートへのアクセスは、メッセージサーバによって実現している。OIDで参照するポートをグローバルポートと呼ぶ。

OIDはシステムがポートに対して動的に割り当てる名前であるため、プログラマはプログラム作成時に通信相手のポートのOIDを知ることができない。この問題を解消するため、Orionではポートに対して任意の文字列でユーザ定義名（UDN:User Defined Name）を付けることができ、さらにその名前を使用してポートにアクセスすることが可能である。通常、アプリケーションプログラムではUDNを使用して通信を行なう。

3．メッセージサーバ

メッセージサーバは、ユーザプロセスや他のサーバプロセスから送られた、ポートの生成／消去要求や、メッセージ配送要求等に対してサービスを行う。またグローバルポートの実現も、メッセージサーバがOIDをローカルポートにマッピングすることによって行う。

メッセージサーバの構成は図1の様になっており、1つのリクエスト受付プロセスと複数のサービス実行プロセスから成っている。

受付プロセスは、受信したリクエストメッセージをサービス実行プロセスに引き渡すために、共有メモリ上に設けたキューを使用する。

サービス実行プロセスでは、渡されたリクエストメッセージを解析し、要求された処理を行なう。メッセージサーバ内にはポートを管理するためのポートテーブルがあり、サービスを行なう際にはこれを参照／更新する。このためにポートテーブルはサービス実行プロセス間で共有する。

メッセージサーバへのリクエストはリクエストコードとそのコードに特有のパラメータで構成するパケットを受け渡すことで行う。パケットの組立及びパケットの送信処理はプログラムにリンクしたMSライブラリで行う。従ってユーザは関数呼び出し形式でメッセージサーバに対するリクエストを送ることができる。

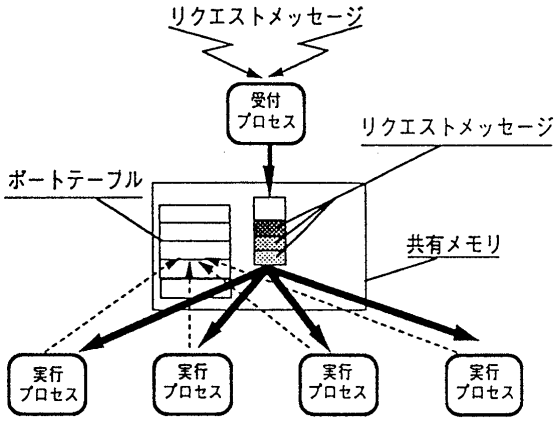

図1　メッセージサーバの構成

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

4．位置情報のキャッシング

あるプロセス（AP）が、グローバルポートにアクセスする際の処理の流れは次のようになる。

1. APが同じノード上のメッセージサーバ(ms1)に対してグローバルポートへのアクセス要求を送る。
2. ms1はネームサーバに対してグローバルポートの位置情報の問い合わせを行う。
3. ms1はグローバルポートの存在するノード上のメッセージサーバ(ms2)に対してアクセス要求を送る。
4. ms2がグローバルポートへの直接のアクセスを行う。

このうち２の問い合わせ処理には、

a. メッセージサーバ - ネームサーバ間のローカルな通信が発生する。
b. 位置情報の一貫性制御のためにネームサーバは全ノード上のネームサーバのロックを取る必要がある。

といった問題があり、これらが通信性能を低下させる原因となっている。bについてはネームサーバが一貫性の強さの概念を取り入れることで解決を試みている。ここではaの問題の解決法について述べる。

アプリケーションの立場からポートの使用を考えてみると、科学技術計算の分野でよく用いられるSPMD方式では通信先は固定的であり、グローバルポートの生成や消去、もしくは移送といった処理を演算実行中に行うことはほとんど無い。この種のアプリケーションの場合、一旦得られたグローバルポートの位置情報をその後の通信で繰り返し使用すること、即ちキャッシングが有効な手段となる（図２）。キャッシングを行うことで、同一グローバルポートに対するアクセスでは、ネームサーバへの問い合わせ処理は最初の１回のみですむため、ローカル通信回数を減少させることができる。

得られた位置情報は、メッセージサーバ内部のポートテーブルに格納し、キャッシュ情報であることのフラグを立てておく。その後の同一グローバルポートへのアクセスの際には、ポートテーブルを検索することにより位置情報を得ることができる。また、グローバルポートが消去された場合には、全ノード上のメッセージサーバに対して当該グローバルポートのキャッシュ情報を無効化するメッセージをブロードキャストする。

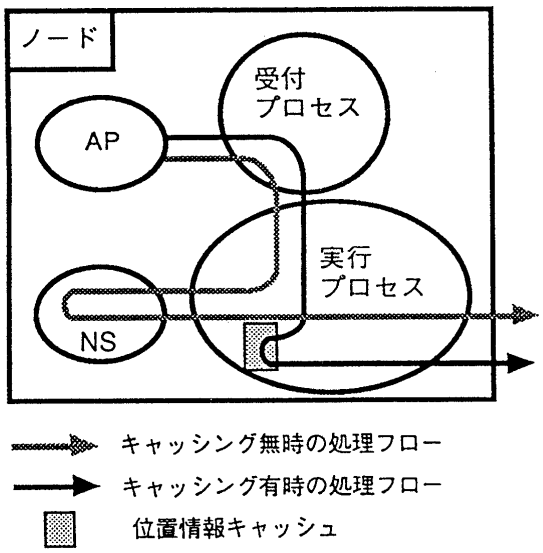

図２ 位置情報のキャッシング

5．性能測定

ここでは、メッセージサーバが位置情報のキャッシングを行なった場合と、行なわない場合との実測結果を示す。

使用した測定プログラムは、ノードAに存在するプロセスがグローバルポートport-Bに4Kbytesのメッセージデータを複数個送信し、ノードBに存在するプロセスがport-Bからそれらのメッセージデータを受け取るものである。port-Bの実体はノードBに作成する。測定する時間は、送り側プロセスが最初のメッセージデータを送り始めてから、最後のデータを送り終わるまでの時間である。測定には、HP/Apollo9000シリーズ720を使用した。送信するメッセージ数は、10, 50, 100と変えて測定を行った。

表１に実測結果を示す。表からも分かるようにキャッシングによってネームサーバへの問い合わせ処理回数を減らすことで、性能が大幅に向上している。特に送信するメッセージ数が多くなると、一つ目のメッセージ送信時に於けるキャッシュミスのコストが相対的に小さくなり、キャッシングの効果が顕著に現われている。

表1 キャッシングの有無による性能比較（秒）

メッセージ数	10	50	100
キャッシング有	2.77	3.50	4.46
キャッシング無	12.7	58.0	116

6．おわりに

Orionのメッセージサーバの実装と位置情報のキャッシングによる高速化について述べた。Orionのプロセス間通信機能は、位置透過な通信、ユーザ定義名による通信相手の指定等の特長を有し、スケーラビリティの高い並列プログラムが容易に記述できる環境を提供する。また、位置透過なプロセス間通信を実現するために必要となるネームサーバへの問い合わせ処理オーバヘッドをキャッシングにより減少した。

今後、通信機能についてはマルチキャスト、同期、ポートのグループ化などの検討を行う予定である。

参考文献

[1] 岩崎, 他, 「分散並列OS「Orion」の試作ーシステムの概要」, 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」の試作ー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.