

分散OS COSMOS 2における プロセス間通信の評価

6P-5

李 在己 相田 仁 齊藤 忠夫

東京大学 工学部

1. まえがき

筆者等の研究室で実装しているCOSMOS 2は、論理機能モジュール（以下、モジュールと略する）による構成と、高速かつ信頼性高い通信を特徴とする統合形分散OSである。従来のOSでの主な機能に対応するモジュールを、高速のLANに結ばれている各ステーション上に適切に配置する。これらのモジュールが相互協同することによって、ユーザにネットワーク透明な環境を提供するシステムを構築することを目指している^[1]。

分散OSにおける性能に関しては、ステーション間で行われるプロセス間通信を能率良く実行することが重要である。この目的で筆者等は、プロセス間通信のためのリモートプロシージャコール(RPC)の諸方式について検討してきた^[2]。この検討においては、従来、RPCの3 ways方式、2 ways方式について主としてUnix上におけるやりとりとしてデータを求めていた。今回は、COSMOS 2の通信環境を実現して、この両方式の性能を求めたので報告する。

2. COSMOS 2のプロセス間通信

Fig. 1に、COSMOS 2におけるクライアントサーバモデルに基づくプロセス間通信のやりとりを示す。モジュール内のクライアントプロセスとサーバプロセスは、通信サブシステムを介して通信を行う。

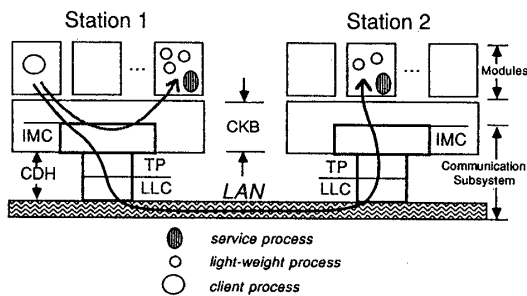


Fig. 1. Inter-process communication based on client-server model between modules.

各ステーションごとに常駐するCKB(COSMOS Kernel Backplane)は、自分のステーションに対す

る初期化とモジュール間のスケジューリング及び通信を受け持つ。特に、モジュール間通信を行うIMC(Inter-Module Communication)層では、ステーション内のモジュール間通信とステーションをまたがったモジュール間の通信を、上位層が同一の扱いをすることができるようにする。これによって、応用レベルでのプロセス間の通信が透明化される。また、ステーションをまたがったモジュール間の通信の場合は、両ステーションの通信デバイスハンドラ(CDH)を介して行う。

モジュールは、マネージャとサーバとデバイスハンドラに大別される。

マネージャには、プロセスマネージャとファイルマネージャがある。前者はシステム内の全プロセスサーバの負荷を統合的に管理し、動的な負荷分散を支援するモジュールである。一方、後者はネームサーバであり、パスネームをシステムで唯一なファイル識別子にマッピングする機能を持つ。

プロセスサーバ、ファイルサーバ、プリントサーバなどのサーバは、各々特定のサービスを提供するモジュールである。

デハイスハンドラは、入出力機器を直接制御するモジュールであり、ディスクデバイスハンドラ、通信デバイスハンドラなどがこれに属する。

デバイスドライバ以外のモジュール間には、クライアントとサーバの関係が形成される。クライアント側のプロセスの要求に応じて、サーバ側のサービスプロセスは、要求ごとに軽いプロセスを生成し、要求に対する処理を行い、その結果をクライアント側のプロセスに渡す。

COSMOS 2でのプロセス間の通信を分類すると以下のとおりである。

- (1) モジュール内のプロセス間通信
- (2) モジュール間のプロセス間通信
 - a. 同一ステーション内
 - b. 異なるステーション間

モジュール内でのプロセス間の通信は、Unixシステムと同一に処理する。一方、モジュール間のプロセス間の通信は、IMC部でステーション内の通信かステーション間の通信かを判別し、専用のメッセージバッファを用いて行う。

3. RPC方式の分類

分散環境でのプロセス間通信に適用されているRPCはFig.2のように三つに分類される。

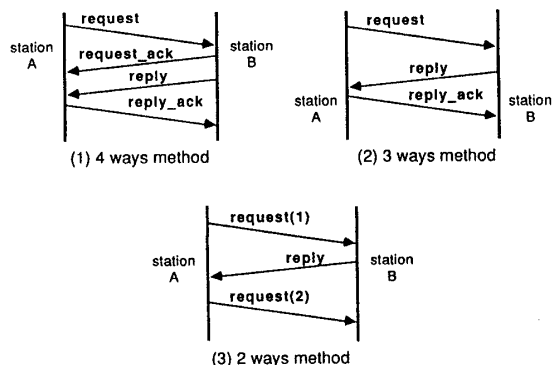


Fig. 2. Classification of RPC methods.

(1) 4 ways 方式

Request に対しては request_ack を, reply に対しては reply_ack を明確に返す. 信頼性の面では一番よいが, その代わりに一回のRPCにつき, 4つのメッセージパッシングが要求され, 性能の面では3 ways方式と2 ways方式より劣る.

(2) 3 ways 方式

Request に対する request_ack の役割を reply に持たせる. Request 後の reply が来るまでの時間間隔が問題になるが, 信頼性と効率の両面を考慮する方式である.

(3) 2 ways 方式

Request と reply に対する明確な応答を行わず, request に対しては reply が, reply に対しては次の reply が応答に代わる. 一回のRPCに2つのメッセージパッシングしか要求されないため, 性能の面では一番よいが, request と reply 間の時間間隔をどう取るかが問題になる.

4. 実測とその評価

COSMOS 2の試作システム上にクライアントサーバモデルを構築し, 2 ways方式と3 ways方式について, メッセージ長が2, 4, 8, 16, 32, 64, 128, 256, 512バイトの場合の応答時間を測定した. Fig.3に, 実測の環境を図示する.

現在, COSMOS 2では信頼性と性能の両面を考慮し, 3 ways方法を採用している. また, メッセージ転送には, 最大64バイトのメッセージバッファを用いる. これには, 各層のヘッダは含まれていない. メッセージのサイズはヘッダを除いて32バイト程度より大きくしておけばよいからである^[2].

Fig.4に2 ways方式と3 ways方式の応答時間の比較を示す. これによると, 64バイトの場合3 ways方式に比べて2 ways方式の方が約25%の応答時間が改善されることが分かる. COSMOS 2の通信環境を適用してない場合の約28%とほぼ同じ比率である.

このように応答時間の改善の比率がすこし下がっ

たことは, 通信処理の時間がその分大きくなったことを意味する. しかし, 両方式の応答時間の差は, COSMOS 2の実測システムでは, 3 ways方式の方が4.2 msecくらい大きい, Unix上で測定した時は, 約2.7 msecである.

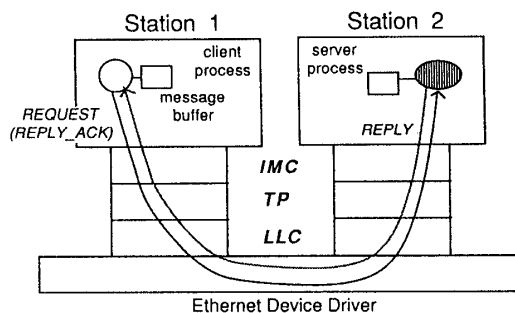


Fig. 3. The environment for measurement.

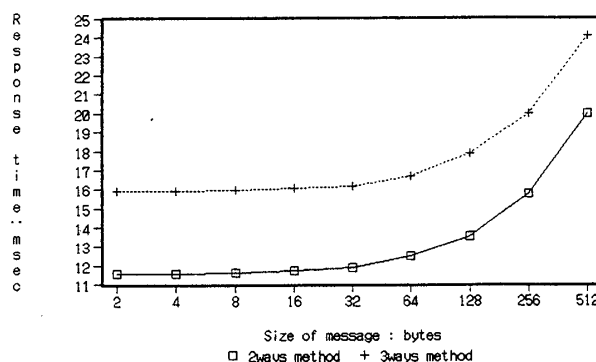


Fig. 4. Comparison of response times between 2ways method and 3ways method.

5. おわりに

本稿では, 分散OS COSMOS 2のプロセス間通信とRPC方式の分類に加え, COSMOS 2試作システム上で行った2 ways方式と3 ways方式のプロセス間通信の応答時間を比較し, 検討した.

分散システムでの通信の場合, 速度と信頼性の間にはトレードオフがあるが, 低エラー率のLANを用いて, 高速なプロセス間通信が要求される分散OSでは2 ways方式が望ましいことを確認した. しかしながら, プロセス間通信における信頼性とプロセス間の容易な同期などの点を考えると, 25%位のオーバーヘッドはあるが, 3 ways方式も分散OSの構築の際に考慮すべきである.

[参考文献]

- 李, 相田, 齊藤, "LANを用いた分散処理方式の通信プロトコル," 電気学会通信研究会, CMN-89-7, Jan. 1989, pp. 55-63.
- 李, 相田, 齊藤, "RPCのためのコマンドレスポンス方式の性能に関する考察," 1989年電子情報通信学会秋季全国大会発表予定.