非同期メッセージ通信機能の実現

3Bb－8

楠　和浩，大谷　治之，中川路　哲男，関口　孝興，小田切　耕司，松田　昇平
三菱電機株式会社

1　はじめに

地理的に離れた複数の部門サーバを、広域網を利用して連携させる企業内／企業間分散処理システムでは、RPC (Remote Procedure Call) を使用した場合の、いわゆる「ブロッキング・オペレーション」による性能劣化を避けるために、非同期メッセージ通信が使用されるようになってきている。非同期メッセージ通信とは、アプリケーションからの要求を一旦キューに格納して処理をアプリケーションに返し、非同期に相手アプリケーションとの通信を実行することにより、通信とアプリケーション処理を分割するものである。

これまでの非同期メッセージ通信は、キュー機能（信頼性確保、順序性確保など）およびノード間の2相コミット機能をメッセージ通信機能が独自にサポートしている。さらに、それらの機能の実行には多くのメモリやCPUを必要としていた。

我々は、クライアント／サーバシステムにおいて、通常サーバに実装されているデータベース機能をキューの実現機能として共有し、さらに2相コミットと比較して通信オーバヘッドの少ない整合性保証プロトコルを実装した非同期メッセージ通信機能を実現した。

2　非同期メッセージ通信機能概要

非同期メッセージ通信機能の概要を、図1に示す。

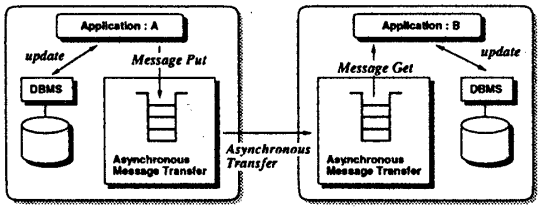

図 1: 非同期メッセージ通信概要

図では、Application A が自ノードの DBMS が管理するデータを更新し、そのデータを Application B に非同期メッセージ機能を通して送信し、Application B 側の DBMS が管理するデータを更新する例を示している。各ノードの非同期メッセージ通信機能は、データ送受信のためのキューを用意しており、アプリケーションはそられのキューに対するデータの enqueue/dequeue によりアプリケーション間通信を実行する。

ここで、高信頼な非同期メッセージ通信には、以下の機能が必要になる。

1. 送信アプリケーションおよび受信アプリケーションの DBMS への更新アクセスと非同期メッセージ機能が提供するキューへのデータの enqueue/dequeue とを同一トランザクションとして管理する機能
2. 非同期メッセージ通信機能間のデータ転送に関するトランザクション制御機能
3. 非同期メッセージ通信間のデータ送信時に発生する通信障害に対する障害回復機能
4. ノード障害に対する障害回復機能

3　実現方式

3.1　実現方針

我々は高信頼な非同期メッセージ通信を実現するために以下の実現方針を設定した。つまり、DBMS が管理する DB 内に仮想的にキューを構成することにより、DBMS が持つノード障害に対する回復機能を、非同期メッセージ転送のノード障害回復機能とする。さらに、アプリケーションに対しては、DBMS へのアクセスとメッセージキューへのアクセスは異なるインタフェースとして提供し、アクセスライブラリでインタフェース変換を実施することで内部的に同じ DBMS にアクセスする。

また、キュー及びキュー内メッセージの信頼性保証を実現するキュー管理（DBMS）と、分散ノード間の転送性能確保と転送信頼性保証を実現する転送管理を分離することにより全体性能向上を図った。

3.2　モジュール構成

図2に実装した非同期メッセージ通信機能のモジュール構成を示す。

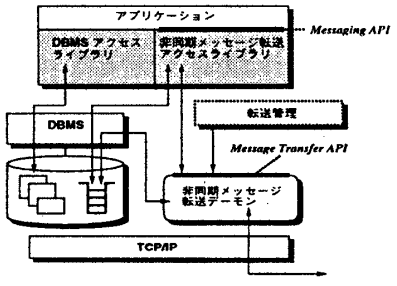

図 2: モジュール構成

非同期メッセージ転送機能に関係するモジュールおよび、それらの概要は以下の通り。

【非同期メッセージ転送アクセスライブラリ】 アプリケーションに対して Messaging API を提供する。さらに、Messaging API を仮想的に用意した非同期メッセージ転送用キュー（DBMS のテーブル）へのアクセスコード（SQL 文）に変換する。

【仮想キュー】 DBMS が管理するデータベース上に仮想的なキューを構成する。

A Development of Asynchronous Message Transfer System
Kazuhiro KUSUNOKI, Haruyuki OHTANI, Tetsuo NAKAKAWAJI, Takaoki SEKIGUCHI, Kohji ODAGIRI, Shohei MATSUDA
MITSUBISHI ELECTRIC CORPORATION

【非同期メッセージ転送デーモン】 アプリケーションまたは転送管理に対して Message Transfer API を提供する。これにより、アプリケーションが実際の転送を制御したり、回線の状態によって転送管理が最適なメッセージ転送を実現可能となる。

【転送管理】 ノード間のデータ転送契機を管理する。

3.3 実現課題および実現方式

3.3.1 メッセージ転送契機

アプリケーションが仮想キューに対して enqueue したメッセージを転送するタイミングには、(1) メッセージを enqueue した直後に転送を開始する即時転送と、(2) 定義パラメータ（タイマやキュー長など）が閾値に達した場合に転送を開始する非同期転送が考えられる。

これまでの非同期メッセージ通信では、(2) の非同期転送だけをサポートしている。しかし、業務アプリケーションによっては、業務ロジックは転送と非同期に動作したいが、転送契機は制御したい、という要求がある。

したがって、今回の実装では、Messaging API として即時転送を指定するオプションを用意し、非同期メッセージ転送ライブラリが即時転送を解釈して、仮想キューに対するメッセージの enqueue およびそのコミットメント処理の後で非同期メッセージ転送デーモンに対する転送指示を出すことで即時転送を実現した。

3.3.2 キュー内メッセージ順序制御

enqueue されたメッセージは enqueue された順番で転送されなければならない。キューを DBMS が管理するテーブルとして実現した場合順序を制御するキーが必要になる。その方法としては、(1) アプリケーションが転送しようとするメッセージの構造を非同期メッセージ転送機能が解釈し、それに応じたテーブル構造とし、順序制御キーはアプリケーションが指定する方法と、(2) 非同期メッセージ転送機能は enqueue されるメッセージの構造を理解せず（つまり単なる binary データとして扱う）、内部的に順序制御キーを付加する方法が考えられる。(1) の方法ではキューの定義および管理が複雑になり汎用性／拡張性がなくなるため、DBMS が用意する順序番号を利用し (2) の方法により実装を行なった。

3.3.3 メッセージ転送制御

分散ノード間のメッセージ転送は、(1) 送信元ノードの仮想キューからのメッセージの dequeue、(2) 送信先ノードへの転送、(3) 送信先ノードでの仮想キューへのメッセージの enqueue という 3 つのフェーズが必要となる。信頼性を保証するためには、これらをひとつのトランザクションとして取り扱い、結果として分散環境での 2 相コミット処理が使用されることが多い。しかしながら非同期メッセージ通信における転送ノード間は、常に高速 LAN による接続形態を取るとは限らず、広域網を使用した場合も想定され、その様な分散環境での 2 相コミット処理は性能上ボトルネックになる可能性がある。

そこで、送信側非同期メッセージ通信機能の DB アクセスと受信側非同期メッセージ通信機能の DB アクセスを同一トランザクションとして処理しない以下のプロトコルを使用して性能向上を図った。

まず、アプリケーションから渡されたメッセージを、制御データと共に管理するメッセージ管理テーブルと、通信障害に対応するための回復管理テーブルを定義した。

メッセージ管理テーブルは、制御データとしてメッセージ識別子、および転送状態フラグを持つ。メッセージ識別子はメッセージ送信側で生成されメッセージと共に受信側に送信される。回復管理テーブルでは、現在までに受信したメッセージのメッセージ識別子が管理される。

送信側および受信側の非同期転送デーモン間の動作概要を図 3に示す。

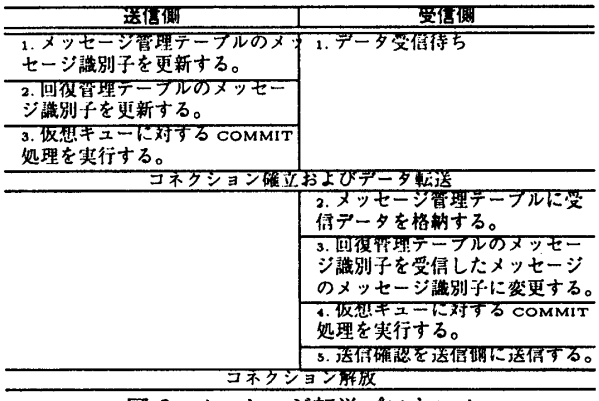

図 3: メッセージ転送プロトコル

3.3.4 障害回復制御

送信側非同期メッセージ通信機能の DB アクセスと、受信側非同期メッセージ通信機能の DB アクセスを同一トランザクションとして処理しないために、それに対応した障害回復制御方式が必要になる。通信障害及びノード障害に対するリカバリ処理の概要を示す。リカバリ動作は、メッセージ管理テーブルの転送状態フラグと、回復管理テーブルのメッセージ識別子の状態により実行される。次の 3 つの場合が存在する。

1. 転送状態フラグ（送信側）が OFF の場合。ノード間のリカバリ動作は発生しない。
2. 転送状態フラグ（送信側）が ON で、ノード間の回復管理テーブルのメッセージ識別子に整合性がない場合 (2-1) とある場合 (2-2)。

図 4 に、(2-1) の場合のリカバリプロトコルを示す。(2-2) の場合には、リカバリ対象メッセージの送信から送信完了までの手順が削除される。

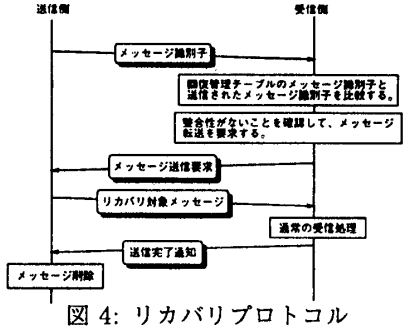

図 4: リカバリプロトコル

4 おわりに

クライアント／サーバシステムにおいて通常サーバに実装されているデータベース機能をキューの実現機能として共用し、広域網によるサーバ間連携を考慮した通信オーバヘッドの少ない整合性保証プロトコルを実装した非同期メッセージ通信機能を実装した。

今後は、通信効率（速度、転送タイミング）の最適化を行なう予定である。