計算機アーキテクチャ　96－19
（1992. 10. 23）

OSI CCRを用いたUNIX用分散トランザクション処理方式

鈴木信雄　　脇　英世
東京電機大学
東京都千代田区神田錦町2-2

あらまし　分散ネットワークにおける基本的な機能に分散トランザクション処理機能がある。現在、この機能を実現するシステムが多く開発され利用されているが、概念や用語等に統一がなく、アプリケーション開発者の負担となっている。一方、このような分散ネットワーク環境における通信方式の国際標準としてOSIがある。OSIは国際標準であることから概念や用語が一般化され、将来の分散ネットワーク環境の共通的基盤を提供するものである。本稿では、汎用的な分散トランザクション処理機能をOSI標準のCCRを用いて実現するための方式について考察し、UNIX上で実装した結果について報告する。また、本実装においては、近年注目されているスレッドを用いて、分散トランザクション処理機能を提供するサーバプロセスを構築した。本稿では、その構築法についても報告する。

和文キーワード　分散トランザクション　マルチスレッド環境

Distributed Transaction Processing Methodologies using OSI CCR for UNIX

Nobuo SUZUKI, Hideyo WAKI
Tokyo Denki University
Kanda-Nishiki-Cho 2-2, Chiyoda-ku, Tokyo 101, Japan

Abstract　Distributed transaction processing is a basic facility for distributed networks. Although there are many systems which implement such facilities, those systems are not consistent. This situation causes application developpers to be responsible for learning many systems. OSI (Open Systems Interconnection) can offer generalization of distributed transaction processing concepts and provides common basis. This paper proposes a method for distributed transaction processing using OSI CCR (Commitment, Concurrency and Recovery) in the UNIX environment. It also describes the construnction of a multi-threaded server.

英文 key words　distributed transaction　multi-threaded server

1. はじめに

現在、我々を取り巻くコンピュータ環境では、ほとんどのコンピュータはそれら単独で稼働しているのではなく、異なる機種を含む様々なコンピュータがLAN(Local Area Network)等のネットワークに接続されて利用されるのが一般的である。そのため、最近では以前にも増してネットワーク機能の充実が重要となってきた。

このようなネットワーク機能の中で、分散配置された資源の更新などに有効な応用に分散トランザクション処理がある。分散トランザクション処理の機能は分散ネットワーク環境での基本的な機能であるにもかかわらず、現在、各社各様のシステムが開発されており、開発システムが変わる度にアプリケーション開発者にとっては再学習の負担が必要となってしまう。

一方、異機種間の相互接続を目的としたOSI(開放型システム間相互接続)は、下位の通信環境の整備がほぼまとまり、現在は、RDA(リモートデータベース・アクセス)やOSI管理などの上位の応用層に関する標準化作業が活発に行われている。このOSIは、国際標準という性格から概念や用語が一般化され、将来の分散ネットワーク環境の共通的基盤を提供するものである。分散トランザクション処理の基本機能としてもCCR(コミットメント、同時性及び回復制御)[1]が規定されている。そこで、筆者らは、現状のUNIX環境下における汎用的な分散トランザクション処理機能を実現するための手法として、OSI応用層の機能単位であるCCRを用いることを提案し[6]、実装を試みた。本稿では、その実現手法について詳細に報告する。

また、マルチプロセッサの普及と高速化の要請から、UNIXに限らず多くのオペレーティングシステムでスレッドがサポートされている。今回の実装では、マルチスレッド・サーバの実現についても検討及び実装を行っており[6]、この点についても報告する。

2. 分散トランザクション処理とOSI CCR

2.1. 分散トランザクション処理

複数のユーザから利用されるデータベース環境を構築する上において必須となる技術にトランザクションがある。トランザクションは、ACIDと呼ばれる次のような4つの性質を持つ一連の操作を意味し、信頼性の高い資源更新処理などには重要な概念である。[3]

(1)原子性 -atomicity-：トランザクションとして定義された一連の操作は、完全に実行されるか、あるいは、全く実行されないかのいずれかである。

(2)一貫性 -consistency-：トランザクションの実行によってデータは矛盾のない状態から矛盾のない状態へ変化する。

(3)独立性 -isolation-：トランザクション途中の結果は他のトランザクションから参照することはできない。

(4)耐用性 -durability-：トランザクションが完了すると、その結果は永久的なもので、障害等によって失われることはない。

このようなトランザクションを資源がネットワーク上に分散配置された環境で実行するのが分散トランザクション処理である。

この分散トランザクション処理機能は、分散ネットワーク環境での基本的な機能であるにもかかわらず、各社各様のシステムが開発されており、それぞれのシステムにおいて概念や用語が異なっている。このような状況では開発システムが変わる度にアプリケーション開発者にとっては再学習の負担が生じてしまうという問題が出てくる。そこで、筆者らは、以下で述べるOSI CCRを用いることでUNIX上での汎用的な分散トランザクション処理機能の実現手法について検討した。

2.2. OSI CCR

(1)CCRの概要

OSIは、コンピュータの異機種間接続を目的として機種に依存しない通信機能を定義している。前述のような分散トランザクション処理は、分散ネットワーク上のアプリケーションにとって基本的な機能であるため、OSIにおいてもCCRやTP(トランザクション処理)として標準化されている。これら2つの標準のうち、TPは下位レベルでCCRを用い、複数ノードに対するトランザクション・サービスを提供する。これに対しCCRは、一対一のノード間での通信プロ

トコルを規定しており、アプリケーションとして様々な形態のトランザクションを実装することができる。今回の検討では、(a)機能が分散トランザクション処理機能の基本的なものに限られているために柔軟性が高いこと、(b)システム全体のパフォーマンスに影響するプロトコル処理の規模が小さいことを考慮して、CCRを用いることとした。

また、このCCRは、国際標準であることから、異機種間接続に限らず、今後の分散トランザクション処理のアプリケーション構築における共通的な基盤となると考えられる。そのため、このCCRを利用することによって、分散トランザクション処理機能に関する概念や用語の統一を図り、汎用性の高いシステムを構築することができる。

(2)CCRの機能

CCRでは、アプリケーション間で行われるトランザクションを原子動作という名称で規定している。原子動作では、協調して動作するアプリケーションを図1のような原子動作木という木構造の関係として定義する。この木構造の上位と下位の間でCCRプロトコルを交換することによって原子動作を進めていく。また、CCRは、原子動作のコミットメント制御を行うために2相コミットメント・プロトコルを用いている。2相コミットメントでは、原子動作終了時に全ノードに処理が正常に終了したかどうかを尋ね、全てが肯定応答ならばコミットメント、一つでも否定応答があればロールバックの指示を出すことにより、全ノードの資源の一貫性を保証する。

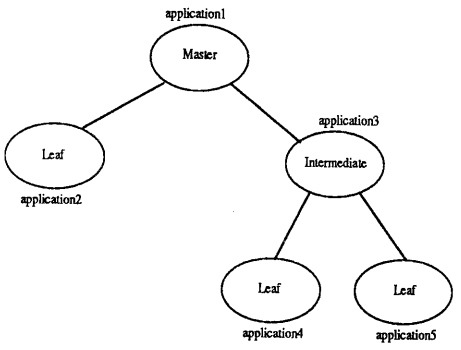

図1　CCRにおける原子動作木

また、原子動作に参加しているノードにおいては、障害が発生しても原子動作を維持できることが必要である。そこで、CCRでは、原子動作データと呼ぶログ情報を保守することによって再起動を行うことができる。

図2にCCRプロトコルのシーケンスを、表1にサービス定義を示す。

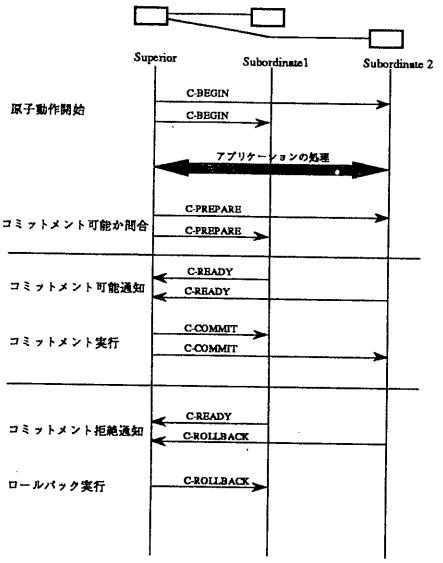

図2　CCRプロトコルのシーケンス

サービス名	機　能	タイプ
C-BEGIN	原子動作開始	応答型
C-PREPARE	資源更新可否の問い合わせ	非応答型
C-READY	資源更新可の通知	非応答型
C-COMMIT	資源更新の指示	応答型
C-ROLLBACK	資源状態の復帰を指示	応答型
C-RECOVER	原子動作の再開を指示	応答型

表1　CCRのサービス定義一覧

3. 実現手法

3.1. CCRの実装範囲

CCRでは、主に2相コミットメント機構に基づくコミットメント制御に関する機能を規定し

ており、同時性制御や回復制御の実現については、ほとんどが実装者に任されている。今回試作したシステムでは、CCRの中心的な機能であるコミットメント制御機能及び同時性制御機能を実現している。同時性制御機能としては、UNIXで提供されているロック機構を用いて資源(ファイル)全体をロックし、他のプロセスからのアクセスから保護することにより実現している。すなわち、今回の実装では、C-BEGIN、C-PREPARE、C-READY、C-COMMIT、C-ROLLBACKの各サービスを実現している。

3.2. 実現の基本方針

(1)分散トランザクション処理機能は独立したサーバとして実現する。

現在、多くのUNIXアプリケーションでは、クライアント/サーバ方式のものが普及している。この方式を用いることにより、サーバの障害がコンピュータシステム全体の停止に至らないという高い耐障害性と、新しい機能の拡張時にはサーバのみの変更だけで行えるという柔軟性を合わせ持つことができる。本システムにおいても分散トランザクション処理機能を実現するモジュールを独立したサーバプロセスとして生成し、クライアントからの要求により処理を行うこととする。

(2)サーバプロセス内はマルチスレッド・サーバとして実現する。

スレッドは、従来のプロセスに比べて生成と消滅に要する時間が短く、マルチプロセッサ環境での完全な並列化を実現できる機構である。一般に、トランザクション処理においては、処理時間の比較的短い資源更新要求が短時間の内に多数発生するような状況が考えられる。このような状況に対してスレッドは特に有効であり、処理時間の短縮を図ることができる。そこで、本システムにおいては、分散トランザクション処理機能を実現するモジュールをマルチスレッド・サーバとして実現することにより、システム全体のパフォーマンスを向上させることとする。

(3)トランザクションサービス・インタフェースはCCRに従った形式とする。

本実装では、分散トランザクション処理のためのノード間のプロトコルとしてCCRを用いているが、アプリケーション開発者が実際に開発するクライアントやサーバ・モジュールにおけるトランザクションサービス・インタフェースについてもCCRに従った形式とする。これにより、概念や用語だけでなく、API(アプリケーション・インタフェース)としての汎用化も実現することができる。表2に各モジュールで使用するサービスの一覧を示す。

サービス名	機　能
TransOpen	トランザクション開始指示
TransOpenSub	トランザクションの他ノードへの拡張指示
Begin	原子動作の開始指示
TransClose	トランザクション終了指示
Prepare	資源更新可否の問い合わせ指示
Ready	資源更新可の通知
Commit	コミットの指示
DoneCommit	コミット終了通知
Rollback	ロールバックの指示/通知

表2　トランザクション・サービス一覧

(4)下位の通信環境にはTCP/IPを用いる。

本来、CCR等のOSI応用層機能は7層から構成されるOSI参照モデルに従ったネットワーク上に実現されるべきものである。しかし、今だOSIは普及の過渡期にあり、現実にはその仕様を満足する分散ネットワーク環境を利用することは困難である。そこで、現在、業界標準となっているTCP/IP (Transmission Control Protocol/Internet Protocol) を用いることで下位層の汎用化を図ることとする。すなわち、異なるノード間の通信にはコネクション型のInternetドメインでのソケットを使用し、自ノード内サーバとの通信にはUnixドメインのソケットを用いることとする。

(5)伝送されるプロトコル・データの形式はASN.1形式に従う。

OSI環境においては、伝送されるプロトコル・データはASN.1(Abstract Syntax Notation One)を用いて構造が定義されている。CCRも例外ではなく、各CCRプロトコル・データはASN.1によって定義されている。今回の実装で

は下位層にTCP/IPを用いているが、将来のOSI環境への容易な移行を促すために、実際に伝送されるプロトコル・データはASN.1に従った定義をBER(Basic Encoding Rule)によって符号化したものを用いることとする。

3.3. システム構成

今回実現したシステムは、Sunワークステーション上のSunOS4.1を用いて構築されており、そのプロセス構成は図3に示すとおりである。また、以下に各プロセスにおける機能の概要を示す。

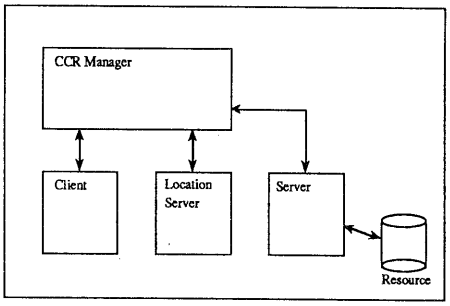

図3　プロセス構成

(1)Client: 分散トランザクション処理機能を利用するユーザプロセスであり、CCR Managerプロセスに対してトランザクションの開始と終了、及び資源更新等の要求を行う。

(2)CCR Manager: 分散トランザクション処理機能を提供するプロセスで、各ノード毎に存在する。本プロセスは、トランザクションの進行に必要な他ノードとのCCRプロトコル・データの交換や複数コネクションの調停機能等を有する。また、後に述べるようにマルチスレッド・サーバとして実現されており、パフォーマンスの向上を図っている。

(3)Location Server: 他ノードとの通信時の位置透過性を保証するためのプロセスで各ノード毎に存在する。このプロセスを利用することにより、Clientプロセスからは、必要な資源を管理しているサーバのアドレスは知る必要がなく、サーバの名称のみを指定するだけで、該当するサーバと通信することができる。

(4)Server:　実際の資源更新処理を行うプロセス。CCR Mangerプロセスからのメッセージに従ってコミットメント処理を進行させると共に、UNIXの提供するロック機構により資源の同時性を保証する。

4. マルチスレッド環境の実現

スレッドは、同一プロセス内で複数の論理的な制御の流れを実行させることができる機構である。これは、複数クライアントに対するサービスを行うサーバなどに有効で、処理の高速化を図ることができる。今回の実装では、分散トランザクション処理機能を実現しているCCR Managerプロセスにおいてこの機構を利用している。ここで、マルチスレッド環境における通信アプリケーションを構築する上で問題となるのが、通信コネクション資源の管理である。通常、通信コネクションはプロセスが管理する資源であるためにスレッドが管理主体となるような機構が必要となる。

4.1. CCR Managerプロセス内スレッドの構成

SunOS4.1ではSunLWP(Lightweight process)ライブラリ[4]というユーザレベルのスレッド機構が提供されており、本システムではこれを使用している。図4にCCR Managerプロセス内でのスレッドの構成を示す。

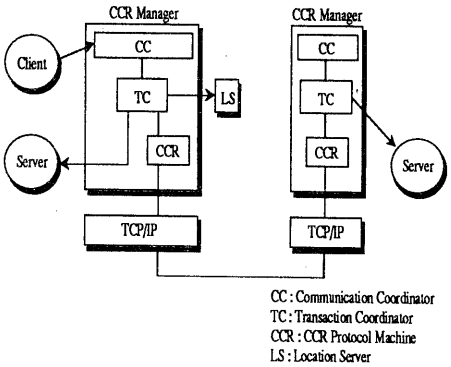

図4　CCR Manager内スレッドの構成

(1)Communication Coordinator: Clientプロセスからの要求を受けてトランザクション毎にTransaction Coordinatorスレッドを生成する。また、複数ノードに渡るコネクション全体の管理を行

う。すなわち、Clientプロセスや他ノードからのメッセージを非同期に受信し、そのメッセージに対応したトランザクションを管理しているTransaction Coordinatorスレッドへメッセージ受信を通知する。

(2)Transaction Coordinator: CCRプロトコル機械の上位に位置し、トランザクション毎に生成される。トランザクションが他のノードに渡る場合は、相手ノードのCommunication Coordinatorスレッドとコネクションを確立し、CCR Protocol Machineスレッドを生成する。以後は、Clientプロセスからの要求をCCRサービスへマッピングし、CCR Protocol Machineスレッドへ渡す。また、一つのトランザクションに参加している複数コネクション間の調停を行い、各Serverプロセスとの間で2相コミットメントの各段階に対応したメッセージの送受信を行う。

(3)CCR Protocol Machine: CCRプロトコルは単一のコネクション上のプロトコルであるため、CCRプロトコル機械を実現している本スレッドは、接続するノード毎に生成される。Transaction CoordinatorスレッドからのCCRサービスを受けてCCRプロトコル・データを生成し、コネクションを通して相手CCR Protocol Machineスレッドへ送信する。

4.2. マルチスレッド環境でのコネクション管理手法

UNIX上では、個々のコネクションはソケットシステムコールによって提供される識別子によって管理される。この識別子はプロセス単位で管理されるもので、複数のスレッドからは共有情報として扱われる。そのために、一つのスレッドが特定のコネクションを使用しようとする場合、スレッドとコネクションの対応を取る必要がある。

この問題に対処するために、本システムでは図5のような手法を用いた。構成要素としては次のようなものがある。

(1)transaction wait queue: Communication Coordinatorスレッドは、Clientプロセスからのコネクション確立要求に対してacceptを行った後、このqueueへソケットを接続し、トランザクション開始要求を待つ。

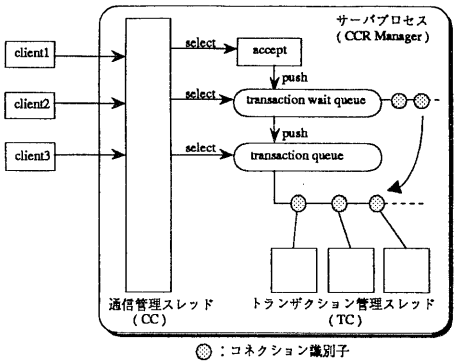

図5 マルチスレッド環境

(2)transaction queue: Communication Coordinatorスレッドは、Clientプロセスからのトランザクション開始要求を受信すると、transaction queue内に新規にトランザクション管理テーブルを生成する。次に、transaction wait queueから該当するソケットを外し、そのテーブルへ保存する。また、Serverプロセスの存在する他ノードへのトランザクション拡張時も、この管理テーブル内にソケットを保存する。その後は、管理テーブル内のソケットをselectシステムコールによりチェックすることでメッセージ受信を検出し、対応するTransaction Coordinatorスレッドへ通知する。

5. 実行シーケンス例

図6のような構成で本システムを実行した場合のシーケンスを図7に示す。この例では、クライアントと同一のノードと、他のノードの2つのサーバに対して資源更新を行い、正常にコミットした場合を示している。

まず、Clientプロセスが"TransOpen"メッセージによってCCR Managerプロセス(CM)へトランザクションの開始を伝える。CMは、これを受けて、トランザクションに参加する各ノードへ"TransOpenSub"メッセージによりトランザクションの開始を指示する。次にC-BEGINメッセージを各ノードへ送信し、原子動作の開始を伝える。ここでCMは新たなトランザクション識別子を定義してClientへ"TransOpenAck"により通知する。以後は、Clientから各Serverに対して更新処理が行われる。

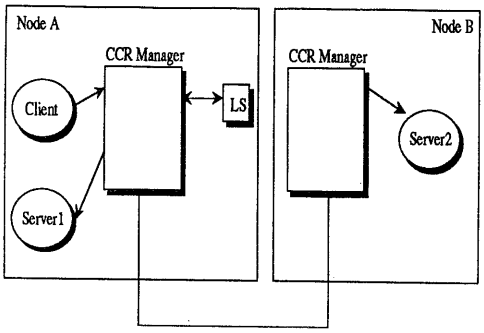

図6　システム構成例

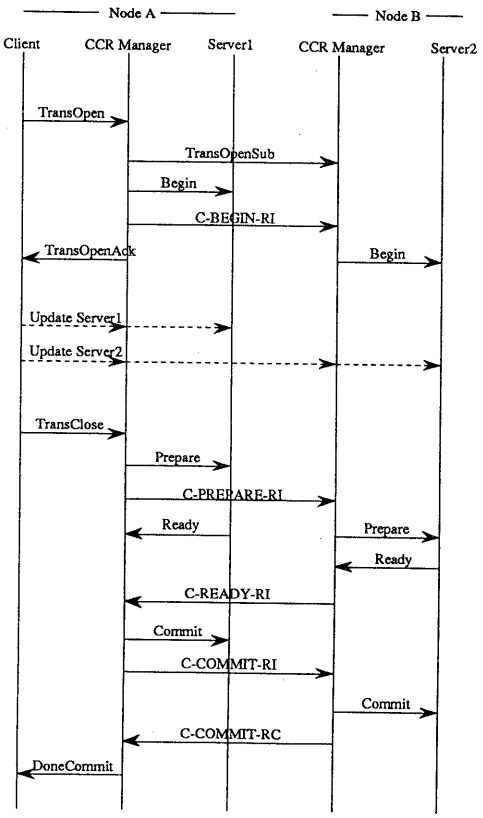

図7　シーケンス例

トランザクションの終了時には、Clientから"TransClose"メッセージによってCMに通知される。CMは2相コミットメント・プロトコルに対応するCCRプロトコルを各Serverの存在するノード内のCMと通信することによってコミットかロールバックかを決定する。コミットであれば"DoneCommit"が、ロールバックであれば"Rollback"がClientへ送信される。

6. 考察

(1)分散ネットワーク環境における国際標準であるOSIの機能単位であるCCRを用いることによって、汎用的な分散トランザクション処理機能を実現することができた。また、マルチスレッド環境でのコネクション管理手法を工夫することにより効率的な分散トランザクション処理サーバを構築することができた。

(2)本システムでは、下位層としてTCP/IPを用いているが、ネットワークで伝送されるCCRプロトコル・データはASN.1の定義をBERによって符号化されたオクテット列である。そこで、下位層をOSI ACSE(Association Control Service Element)以下のOSI環境に移行することにより、比較的容易にOSI環境へ移行することが可能である。

(3)本システムでは、CCRの持つ機能の内、コミットメント制御機能と同時性制御機能を実装した。しかし、回復制御機能もCCRの持つ重要な機能の一つであり、分散トランザクション処理においては必須の機能である。そこで、この回復制御機能についても今後実現の検討を進めていく必要がある。実現に当たっては、コネクション資源と同じくログファイル等の資源もプロセスが管理主体となるため、マルチスレッド環境での管理手法について考慮しなければならない。

(4)OSI CCRは、OSI応用層の一機能単位であり、OSI参照モデルに従ったネットワーク環境を前提として開発されたものである。しかし、今回の実装により、既存のTCP/IPネットワーク上でも実用的であることが示された。これは、CCRのプロトコルが比較的小規模であること、及び、トランスポート層に相当するTCP/IPの上位に実装したことなどがその理由として挙げられる。

7. おわりに

本稿では、OSI CCRを用いたUNIX上での分

散トランザクション処理システムの実現法について述べた。本システムは、国際標準であるOSI CCRを用いていることから、アプリケーション開発者に対する概念や用語等の統一を図ることができるという利点を持つ。また、トランザクション処理機能を提供するサーバプロセスをマルチスレッドサーバとし、この環境でのコネクション管理手法を工夫することによって効率的なサーバが実現できた。

参考文献

[1]ISO/IEC:"IS 9804/9805 CCR",1990
[2]W.R.Stevens:"UNIX Network Programming",Prentice Hall,1991
[3]E.V.Krishnamurthy,V.K.Murthy:"Transaction Processing Systems",Prentice Hall,1991
[4]Sun Microsystems:"Programming Utilities and Libraries,Lightweight Processes",March 1990
[5]Naser S. Barghouti, et. al.:"Concurrency Control in Advanced Database Applications",ACM Computing Surveys, Vol.23, No.3, Sep. 1991
[6]鈴木,脇:"オープン環境を指向した分散トランザクション機能の構成法",電子情報通信学会1992年秋季大会,D-67,1992.9
[7]鈴木,脇:"OSI CCRを用いたUNIX用分散トランザクション処理機能の実現",情報処理学会第45回全国大会,4R-05,1992.10