簡易トランザクション管理機能つき半導体内部ファイルシステム

A semiconductor internal file system with the simple transaction manater

大本　敏雅
Toshimasa OMOTO

ふきはつソフト合資会社　代表
（〒226-0006 横浜市緑区白山二丁目４－１９－２０７　E-mail: futa@mtc.biglobe.ne.jp ）

ABSTRACT. A client must join a distributed transaction to do the realization of the electronic commerce with the reliability that it went through the network and the control system. But, still client's lack of ability becomes bottleneck, and it isn't actually being done. That necessary function was realized, and a bottleneck was dissolved. That is the semiconductor internal file system that a resources manager was combined with the simple transaction manager for the embeded target.

１．背景

IT ビジネスの進展のため一般ユーザが簡単に使える情報家電や携帯電話などによる非 PC 系のクライアントによる確実な電子商取引が求められている。

しかし現状では通信の切断や遅延、クライアントの瞬停に対して自動回復されないため、即時ネットワーク決済を伴う電子商取引はほとんど実現されていない。現金決済を別に行う、代引き、プリペイドやコンビニ決済、またクレジット決済や銀行振込を別に行う通信販売形式がほとんどである。

携帯電話を使った小額決済システムにおいても、なぜか携帯電話網側を利用した決済は実現されておらず、コンビニや自動販売機の側のネットワークを利用した決済となっている。

一方、さまざまな分散協調システムが提案され試作されてきたが、ひとつも実用の域まで達しなかった。これらにはクライアントに ACID 特性を持つ不揮発性記憶域が存在しないため分散トランザクションに加入できないという共通の原因がある。

従来から故障や通信障害に対して間違った結果を出力、永久化しないための技術として ACID 特性を持つトランザクションを実行する方法が確立されてきた。古くは単一のサーバがすべての機能を ACID に実行する集中型のトランザクションシステムが実現された。近年では自立分散処理に向けた分散トランザクションの方法も実用化された。クライアントを分散トランザクションに含めれば、より確実に複雑な処理を実現できる。

しかし PC を含めたインターネットや携帯電話網のクライアントは構成に不備があるため分散トランザクションに加入できない。

以下にクライアントが分散トランザクションに加入できない現状の問題を段階的に分類して示す。若い項番ほど制限が強く、それ以降の問題をすべて抱えている。

（1） 全くトランザクション処理をしていない形態

コマースサイトがユーザに見積もりの自動化だけを行い、カタログ雑誌の代わりに Web を使い、通信方法に E メールを加えた通信販売形態。単にユーザと販売店の双方が人手で従来型の決済手段と連絡手段を使った通信販売を行う形態。

この自動化されない決済例が現金決済を別に行うプリペイドや代引き、コンビニ決済、またクレジット決済や銀行振込を人手で個別に行う方法である。たとえばコンビニや宅配業者が商品受け渡し時に、専用端末でクレジットやデビット決済する。

この範疇にクレジット番号をコマースサイトに直接、教えて決済する方法も含まれるが、これは（４）-b)の問題を引き起こす。

（2） 単一サイトの集中トランザクション処理

インターネットプロバイダや有力なポータルサイトがあらかじめエンドユーザのクレジット番号などの決済関係の機密情報を保持しておき、そのサイトでだけ電子商取引をする形態。電子商取引を単一サイトに制限する理由とは（４）-b)で示す問題で複数サイトにプライバシーや機密情報が拡散することを防ぐためである。ただし、これは（４）-b)の問題を解決しているわけではなく、その制限内で使っているだけである。該当の単一サイトだけは全面的に信用するしかない。「いつどこで誰が何をいくらで買ったか」をすべて単一サイトに記録される。

また、その単一サイトが用意しているサービスに制限される。インターネットを利用していてもパソコン通信時代と変わらない取引形態になる。サイトが（４）-b)の制限を逆手にとって独占的地位を確保しようとするものという見方もある。

（３）以降の問題は解決しないため、自動化できる決済方法もクレジット決済に限定される。クレジット決済は「売り掛け」方式のため、1 ヵ月後にまとめて決済すればよいので、即時決済のための分散トランザクションに対応する必要が無い。他の決済方法については（１）

と同様に全く自動化されない。
またサービス内容も制限されるため、売り切り制のオンラインコンテンツ販売は実用にならない。月額料金制のオンラインゲームは運用されている。

（３） サーバ間分散トランザクションだけを利用する

クライアントが分散トランザクションに加入していなくてもサーバ側の資源だけを使ってサーバ間の分散トランザクションを実現すれば、その範囲内では中断しても事故にはならない。コマースサーバと決済サーバが分散トランザクションを実現すれば取引と決済のアトミック実行は保障される。このような形態の決済方法としてインターデビットが実現されている。デビット決済の暗証番号を守りながらコマースサーバが金融機関のサーバと即時に連携するためサーバ間分散トランザクションに対応している。しかしクライアントが直接、決済処理をしない方法では、画面上の表示金額と決済した金額が一致しない可能性が残る。
また、クライアント側は障害によってはトランザクションが完結したのかどうか分からない。実際の商取引に置き換えると商品を紛失したり、代金を紛失するような事故にはならないが、商取引が成立したのかしなかったのか判断できない状態がありうる。
したがって、このような決済方法においてもクライアントが分散トランザクションに加入できたほうが望ましい。
また、この方法では取引自体にローカル資源を消費する（４）-a)の形態の業務は実現できない。そのために実際に取引されている商品は金融商品が多いようである。

（４） クライアントを分散トランザクションに含める

この形態で初めて以下の機能が実現できる。

a） クライアント資源の更新を伴う分散処理

ここでのクライアントの資源とは不揮発性記憶域のことであり、一般にファイルのことである。複数のクライアントや複数のサーバの間で複数のファイルを更新する処理を ACID な分散トランザクションとして実行する必要がある。すべての処理が可能なときだけ、すべてのノードのすべてのファイルが同期して更新されなければならない。何らかの理由により、ひとつでも失敗する操作があれば、すべてのノードのすべてのファイルは全く更新されてはならない。
もし分散トランザクションに加入せずにクライアント側の資源を更新すると、障害により致命的な事故を起こすため、ネットワーク上の処理に使えない。クライアント内のファイル間の状態も矛盾するし、ノード間の状態も矛盾する。
たとえばファイルのダウンロードがローカル資源の更新に当たるのでオンラインコンテンツ販売の自動化が実現できない。販売されたのに決済されないか、決済されたのに販売されないまま中断し、放置されることがありうる。
（３）以前の現状のコンテンツ販売では決済が先に行われ「ダウンロードが中断したら 24 時間以内に何度でもユーザ自身の操作で同一コンテンツをダウンロードしなおしてください」という方法が多い。人間が異常を察知し不定形の確認・回復作業をする必要があるため、利用者が限られる。
次にクライアントに記憶した電子マネーでネットワーク決済しようとして中断した例を考えると、上記の問題に加え、電子マネー自体が分散トランザクションの資源になるので、取引の成立にかかわらず電子マネーが紛失する事故がありえる。また、そもそも電子マネーのネットワーク側からの補充の際に異常が起こると紛失する危険がある。したがって現状でネットワーク決済を行える電子マネーとは、プリペイド式のワンタイム簡易電子マネーか、（３）の派生型であるサーバ側ウォレットのネットワーク専用電子マネーだけである。しかし、これらは実は現金決済であるか銀行決済（と同等のもの）に該当し、オンラインでもオフラインでも使える本来の電子マネーとはいえない。

b） 通信資源にかかわる分散処理の問題

表面的にクライアントの資源を使っていない取引でも、クライアントに分散トランザクション能力が無いため、通信処理がトランザクションとして統合されない問題がある。単一トランザクション処理では電子商店サーバとクライアントの間の１つのセッションしかないため、決済にかかわる秘密情報やプライバシー情報もすべて電子商店サーバに蓄積されてしまう問題である。これではネットワーク上でスキミングされても防ぎようが無いため、セキュリティ上の問題を引き起こしている。
「商取引」「決済」「認証」などの個別機能は異なる事業者のサーバによって相互認証されることによって信用できる。クライアント／サーバ間も複数の異なるデータリンクとファイルによって、論理的に別々に管理される必要がある。もし通信障害やノード障害によってトランザクションが中断されたときファイルシステムに ACID 特性がなければ、複数のファイルの状態値が不正な値で中途半端に更新されたまま永久化されてしまい回復が困難になる。またファイルシステムに ACID 特性があっても、このような障害時に一律にロールバックしたのでは、ノード間で状態が不一致になってしまう状態もあるので、２相コミットプロトコルと呼ばれる特別なプロトコルが必要である。

c） 分散実行体系の実現

ユビキタスコンピューティングに向けてさまざまな分散実行体系が提案されてきたが実用化されていない。分散実行体系では多くのノードに処理を分散させ、ひとつの処理を実行させる。したがって、ひとつの障害がすべての処理に波及してしまう。特に全体処理の正当性を保障することすら困難になっている。その個別の理由は上記 a)b)と同様であり、単一ノード内の複数の不揮発性記憶データの状態や、複数ノード間の状態値が整合しなくなるからである。したがって、このような体系ではすべてのノードが分散トランザクションに加入していなくては正当性を保障できない。
あらかじめ配置されたプログラムで処理を分担するだけか、動的にプログラムごと移動するタイプがある。どちらも不揮発性記憶を使う限り分散トランザクション的に実行しなければならない。プログラムを移動させる方法を移動エージェントと呼ぶことがあり、オブジェクト指向用語ではオブジェクトフローと呼ばれる。オブジェクト指向用語で不揮発性記憶を使うときは「オブジェクトの永続化」と呼ばれる。プログラミングスタイルによって言葉遣いが違っても本質は同じなので、このような体系を完成させるには、すべてのノードに分散トランザクションに加入するだけの能力が必要とされる。
分散実行体系によっては不完全ながら２相コミットプロトコルを実装しているものもある。しかし各ノードの不揮発性記憶管理の能力は体系の実装以前の前提条件と

して残されたままなので完成に至っていない。分散実行体系を実装する前に各ノードの能力を高める必要がある。

２．目的

電子商取引や分散実行体系の実現のボトルネックはクライアント個別の能力であることを示した。平成 12 年度ならびに平成 13 年度の未踏ソフトウェア創造事業（プロジェクト名：簡易トランザクション管理機能つき半導体内部ファイルシステム）の目的はこのネックを解消するファイルシステムを構築することである。

分散トランザクションに加入するためには X/OPEN 分散トランザクションモデルで定義される２つの機構を実現する必要がある。１つはトランザクションの操作対象である不揮発性資源管理機構の ACID 特性を満足することである。平たく言えばファイルシステムの ACID 特性を実現する必要がある。一般的に、一度でも矛盾したままの不揮発性記憶を引き継いでしまうと、その後でいくら正常なトランザクションを実行しても二度と正常な状態には復帰できない。したがって、複数のファイルの複数の更新をまとめてアトミックに実行できるファイルシステムが必要である。

２つ目は分散トランザクション管理機能を実装することである。分散トランザクション特有のノード間の状態を ACID に保つための管理機能を実現する。この機能は２相コミットプロトコルを終端し、システムダウンをまたがっても回復動作をするために不揮発性の状態値を必要とする。したがってトランザクション管理機能もファイルシステムに統合して実装する。

サーバ側の構成要素や分散実行体系やプロトコルはすでに存在するため、当プロジェクトでは、この部分は開発せず、組み合わせられるように配慮する。ただし標準プロトコルと呼べるものは無く、XML ベースなどで提案、構築中である。

汎用的な分散トランザクションの資源管理として分散データベースを直に使うことは出来ない。分散データベースで簡単にトランザクションを実行できる前提条件とは、内蔵されている専用トランザクション管理をそのまま使う場合である。具体的にはすべてのノードにあらかじめ同じベンダのデータベースパッケージをインストールしておきデータベーススキームも一致させておく必要がある。これはクライアント／サーバ共に不特定多数を相手とする電子商取引に使えない。

たとえば RDB の業界標準プログラムインタフェースとして ODBC や JDBC が規定されているが、これを使ってもプロトコル内部は隠蔽されており、あらかじめインストールした同じパッケージの分散 RDB のときだけ使える。

また電子商取引などの汎用トランザクションは共有データベースの分散更新とは異なる。むしろ共有しないデータを隠蔽しながら連携する必要があり、似て非なる概念である。

分散データベースの内臓トランザクション管理機能を使わず、資源管理としてだけ使うことも出来るが、その場合は簡単に使えるわけではなく、むしろデータベース固有のインタフェースが邪魔になるだけであろう。

たとえば RDB のプログラムインタフェースは SQL ベースになり、RDB の単一行列成分を単一ファイルのように扱わねばならないためファイル名で特定することも出来ない。

したがって当プロジェクトでは標準ファイルシステムを資源管理とし単一ファイルを単位資源とする。プログラムインタフェースは ANCI C 言語ライブラリ準拠とした。この仕様にすることで従来のアプリケーションを利用可能としつつ、軽量クライアントに無駄の無い新規機能の実装を実現した。

分散トランザクション用クライアント側トランザクション管理機能と資源管理機能を統合した内部ファイルシステムとして以下の機能を統合した。

(1) 単なる内部ファイルシステム兼トランザクションの資源としてのファイルシステム。
(2) ローカルトランザクションの実行を保証するトランザクション管理機能。
(3) 分散トランザクションの実行を保証するトランザクション管理機能

そして、軽量クライアントとして簡略化する機能を以下に示す。

(1) データの複製がある分散データベースを目的としないので、複製制御は必要ない。
(2) シングルユーザが同時には１つのトランザクションだけを実行するモデルを前提とする。
(3) ２相コミットプロトコルの調整者の下で参加者だけになり自分の配下に入れ子の参加者を追加しない。
(4) 資源管理の記憶媒体にハードディスクではなく不揮発性半導体メモリを利用する。
(5) SRAM 状の不揮発性メモリの実装を前提とする。それに付加する NOR フラッシュもサポートする。
(6) NOR フラッシュのデバイスドライバはターゲット依存なので利用者に作ってもらう。

SRAM 状の不揮発性メモリとはバッテリバックアップされた SRAM、強誘電体 RAM（FeRAM）、磁気抵抗 RAM（MRAM）などソフトウェア的に I/O を伴わないものである。

NOR フラッシュの不揮発性メモリとは「書き出し」と「消去」が I/O になり、「読み込み」は I/O にならないタイプのフラッシュメモリを想定している。

μITRON などのリアルタイム OS 上のスレッドセイフなライブラリとして実装する。32bit CPU を前提とする。アドレス空間は物理アドレスか、それと同等のページ固定アドレスを前提とする。組み込みシステム特有の制約を解決するために、かなりの最適化を施す。プログラムサイズは全体で 64KB 未満を目指す。

３．外部仕様

表 1 にトランザクション管理(TM)と資源管理(RM)の

プログラムインタフェースの概要を示す。
表 1 は両 API を実行順に並べたイメージである。アプリケーションが同じ RM インタフェースを使っても TM インタフェースで囲まれていなければ単なるファイル I/O であり、トランザクションとして管理されない。
さらに TM インタフェースのうち prepare_trans を使った系列は分散トランザクションとして管理される。

表１ 提供するプログラムインタフェースの概要

TMインタフェース	RMインタフェース
	fopen
start_trans	
	fread
	fwrite
	fseek
	ftell
	fflush
	fgetc
	fgets
	fputc
	fputs
prepare_trans	
commit_trans	
delayed_commit_trans	
abort_trans	
	fclose
	remove

RM は ANCI C 言語ライブラリ準拠だがオープンモードにテキストモードとバイナリモードの区別は無く、全てバイナリモードと同様に働く。

RM と TM の他に SRAM 状の不揮発性メモリを対象とした可変長メモリブロック管理を実装し提供するが、ここではスペースの関係で紹介しない。

４．内部構造

当ライブラリは主に次の三つのモジュールから出来ている。

（1） 不揮発性メモリ管理

トランザクションとは異なる単一操作のアトミック実行だけを保証する SRAM 用不揮発性メモリ管理ライブラリを作成。単一操作とは書き込みを含む生成と消去に相当するもので変更は含まない。
当メモリ管理は可変長不揮発性メモリブロックを割り当て、インデックスを付けて、利用者が情報を検索できるようにする。
この機能や内部プログラムをファイルシステムやトランザクション管理の基盤として使っている。

（2） ファイルシステム（RM）

トランザクションの資源管理となるファイルシステムである。この能力は単に異常時にファイルブロックが乱れないジャーナルファイルシステムではない。その上にトランザクション全体の ACID 動作を実現する。これはトランザクションに含まれる複数のファイルの複数の更新をコミットし、全てまとめて実行するか、ロールバックし、全く実行しないことを制御できるのである。
ただし、このときの更新とは変更ベースである。ファイルの新規作成や削除まではトランザクションに含められない。もしファイルの生成をトランザクションに含めて内容を書いた後にロールバックした場合はサイズ 0 のファイルが作られる。
このファイルシステムは ACID 動作に対応するためシャドウブロック方式で動作している。シャドウブロックは SRAM の固定長ディスクブロックであり、トランザクションログ領域の代わりに使われる。この方法のためにログ領域が節約されディスクキャッシュとしても動作する効率的な実装となる。
ディスクブロックは同一ディスク内では固定長であるがディスク容量によって次のように異なる。これはファイルインデックスが 16bit 値で管理されるためである。

ディスク容量	ブロック長
128MB	2KB
256MB	4KB
512MB	8KB
1GB	16KB
2GB	32KB
4GB＊	64KB

＊最大値だけは十進数の 4000000000

2 種類の半導体メモリに対応しながら同じ外部仕様を維持するため、制約の多い NOR フラッシュが SRAM ファイルの機能を極力制限しないように配慮した。ただし、変更と追加を明確に分ける必要があった。

a） SRAM ファイル

このファイルシステムは SRAM ファイルに最適化してある。NOR フラッシュファイルが有っても無くても関係なく常に最高の性能で実行する。
SRAM ファイルだけを実装するように構成できる。その場合のプログラムサイズは約 44KB と 3 割強低下する。
ターゲットの制約により SRAM の連続領域が採れないときなどに別の領域を複数のディスクドライブとして定義、構成できる。ただし、全てメモリアクセスなので並行動作は出来ない。

b） NOR フラッシュファイル

NOR フラッシュデバイスは書き込み機能に制約が多いため、トランザクション制御用の複雑なデータ構造やバッファは SRAM にとることにした。このため NOR フラッシュファイルの性能も向上している。
ただし、この依存関係があるため NOR フラッシュファイルだけを実装し SRAM ファイル機能を削除することはできない。NOR フラッシュファイルは常に SRAM ファイルと同時にフルセットのプログラムで提供される。プログラムサイズは約 64KB である。
NOR フラッシュデバイスの書き込み、消去は I/O であるためにマルチタスク動作に対応している。別のタスクが別のフラッシュデバイス（ドライブ）を書くときは並行動作する。読み込みはメモリアクセスなので全く並行動作しない。

NOR フラッシュメモリには書き込みと消去の非対称性という特徴があり、固有のメモリ管理が必要になる。
また書き換え消去回数が 10 万回を超えると動作不良が多くなるため、消去回数を平均化し、障害管理をするメモリ管理も必要になる。
結果的に断続的にガベージコレクションを実行しながら全ての領域を均等に使っていく方法をとる。

（３） トランザクション管理（TM）

段階的にファイルシステムの実装に組み込まれていたトランザクション管理用の仕組みを活用して、全ての場合のトランザクションの制御を統合管理する。

a） 通常動作

同時にオープンしているファイルのうちトランザクションを宣言したタスクでオープンしたファイルだけをトランザクション管理する。

同時に実行できるトランザクションは１つに制限する。別のタスクで start_trans が呼ばれたらエラーリターンする。

実行中のトランザクションの状態遷移を管理し正当な状態値を SRAM 状の不揮発性メモリに記憶する。

コミットが指示されたらトランザクション配下の全てのオープンファイルの更新を永久化する。

アボートが支持されたらトランザクション配下の全てのオープンファイルの更新をロールバックする。

b） ダウン後の回復動作

b）1.一般ファイル

トランザクション配下にない一般ファイルはダウン直前の更新を全て反映した状態にする。

b）2.ローカルトランザクション

コミット点に達していないローカルトランザクション配下のファイルは全てロールバックされる。

コミット点に達したファイルは更新を永久化される。

b）3.分散トランザクション

コミット準備中(PREPARED)に達していない分散トランザクション配下のファイルは全てロールバックされる。

コミット準備中(PREPARED)に達しているファイルは状態を引き継いだままアプリケーションサイドに通知する。アプリが２相コミットプロトコルの調整者に問い合わせコミットするときは delayed_commit_trans を呼んで終了する。アボートするときは abort_trans を呼んで終了する。

コミット点に達したファイルは更新を永久化される。

分散トランザクションのコミット準備中(PREPARED)状態の引継ぎについて補足する。これは２相コミットプロトコルの参加者が１相目の問い合わせに対して肯定応答を返した後の状態であり、どんなことがあっても２相目の確定指示を待たなくてはならない。これを２相コミットプロトコルのブロッキングと呼ぶ。当トランザクション管理では上記のようにシステムダウン後も再開始し２相目を催促することが出来る。

この遅延状態がサーバ側の理由で続くことも考えに入れて、トランザクションは２つまで遅延実行できるようにしている。ただし、ファイルが競合するときはファイルオープンがエラーリターンするので、そこで止められる。

５．評価

平成１２年度は主に SRAM ファイルシステムまでを実装し、正常系、準正常系、異常系の検査を行い ACID な機能までを確認した。

平成１３年度は前年度分の改良と NOR フラッシュファイルシステムを含めた全体を実装した。前年と同様の検査も通った。

その結果の平成１３年度末の全体の概略の完成度とは

不揮発性メモリ管理	機能、性能とも OK
SRAM ファイルシステム	機能、性能とも OK
NOR フラッシュファイル	機能 OK、性能 NG

NOR フラッシュメモリの書き込みと消去の非対称性という特徴は、上書きが出来ないため、予想以上に難があり、ガベージコレクションの完成度が低かった。このときのアルゴリズムでは空き領域が少なくなるほど無駄な計算量を費やすものだった。したがって異常系の検査をする前に改造する必要があった。

またフラッシュメモリファイルの追加によってファイルの変更と追加モードの区別による条件が明確になり最適化が可能であることがわかった。

平成１４年暦年中はフラッシュメモリのダーティ消去ブロックの管理アルゴリズムを改良することと、ファイルの変更と追加オープンモードの区別による最適化の変更を行い、異常系の検査をメインに行う予定。来年には商品化したい。

６．まとめ

インターネットや移動体通信網のような接続が保証されない通信環境において、クライアントの不揮発性メモリに状態値を管理することによって、分散トランザクションがクライアントの資源を含めて ACID に実行される。2相コミットプロトコルのブロッキングが起こった場合は、そのまま遅延して回復できる。

その結果、従来は不可能だったコンテンツ販売と即時決済がセキュリティや著作権保護を満たしたまま実行可能になる。

実用化できなかった分散実行体系が実用化でき、息を吹き返す。その結果、高度な産業機械、生産システム、ユビキタスコンピューティングが実現する。

謝辞　当プロジェクトは未踏ソフトウェア創造事業として高田広章プロジェクトマネージャによって採択され IPA の支援により行われた。関係者のご支援に感謝します。