Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

ディスク共用DBクラスタに向けた分散ロックマネジャ

新田 淳[†1,†2] 石川 博[†3]

高性能で高可用な大規模 OLTP システムにおいてよく利用されるディスク共用 DB クラスタシステムにおける，分散ロックマネジャを開発した．特別なロック専用ノードを必要としないように，一般の DB ノード上で稼動するユーザモードのソフトウエアとして実装してある．ロック対象の DB リソースを複数グループに分割し，グループごとにロックマスタを決めることでロック管理を行う．ロックマスタは，ノードの開始・正常終了・異常終了等のイベントに応じて動的に移動する．単一ノード障害に耐えるため，少なくとも 2 ノード上のロックマネジャが，同一のロック情報を保持する．通常ロック処理の通信オーバヘッドを抑えつつ，このロック情報の多重化を実現する方式に最大の特徴がある．本方式は，メインフレーム上の製品として実用化され，金融機関の基幹系などで利用されている．

A Design for Distributed Lock Manager in Shared-Disk DB Cluster

Jun Nitta[†1,†2] and Hiroshi Ishikawa[†3]

We present the design and the implementation of a distributed lock manager for shared-disk DB clusters which are used in highly-scalable and highly-available OLTP systems. This lock manager does not require dedicated hardware but is implemented as a user-mode software running on each DB nodes. Lock resources are devided to multiple groups to each of which a lock master node is assigned. Lock masters migrate among nodes according to events such as node activation and termination. At least 2 nodes hold same lock imformation in their memory simultaneously to endure sigle point of failures while keeping inter-node communication overhead low for ordinary lock operations. A mainframe software product that implements this design are used in various OLTP systems including misson critical online systems for financial institutions.

1. はじめに

本報告では，ディスク共用 DB クラスタシステムにおける分散ロックマネジャの一実現方式を示し，その設計における様々な課題を議論する．ディスク共用 DB クラスタシステムは，ノード数に応じたスケーラブルな性能と高い可用性を実現する有効な手段として，多くの大規模 OLTP システムで用いられている．我々の方式は，メインフレーム向けの製品[1)]として実用化されており，金融機関の基幹系などで使われている．

単一計算機ノードで実現できる限界を超えた性能と可用性を要求する大規模オンラインシステムを構築するには，何らかの形態のクラスタ構成を採用することが必須である．複数ノードで並列処理することによる性能の拡張性と，冗長構成による可用性の向上が見込めるためである．しかし，どのようなクラスタ構成を採るのがよいかは，システムに求められる要件によって変わってくる．参照主体で弱い一貫性制約で十分なシステムの場合は，データと処理モジュールの複製を多数並べる非共用 (shared-nothing) クラスタ構成を採用するのが一般的である．データの可用性は，複数ノードのローカルストレージに DBMS レベルでの複製を持つことで確保される．これに対し，強い ACID 属性が要求される更新処理が多い OLTP システムでは，処理は複製してもよいがデータはできるだけ複製を持たずに集中管理する方が有利であるため，ディスク共用 (shared-disk) クラスタ構成を採用する場合が多い．データの可用性は，ストレージレベルでの RAID による冗長性で保証する．分散システムの CAP 定理[2)] に即して言えば，システム要件の優先度順位が $P(artition-tolerance) > A(vailability) > C(onsistency)$ となる場合は shared-nothing を，$C > A > P$ となる場合は shared-disk を使う傾向が強いということである．

OLTP 向けの応用システムで求められる性能と可用性を満たし，同時にシステム設計と運用負荷を軽くするため，ディスク共用 DB クラスタでは中〜大型計算機を少数並べる構成を採ることが多い．かつては，クラスタ構成を実現するために，各ベンダ固有の I/O 接続アーキテクチャに依存することが多く，コストがかかりがちであったが，FC-SAN の普及により標準的に構成することが可能となった．ストレージデバイスには，大容量のキャッ

†1 静岡大学創造科学技術大学院 自然科学系教育部
Shizuoka University, Graduate School of Science and Technology, Educational Division
†2 株式会社 日立製作所 情報・通信システム社
Hitachi, Ltd., Information and Telecommunication Systems Company
†3 静岡大学情報学部
Faculty of Informatics, Shizuoka University

Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

シュメモリを備えた RAID コントローラ配下に装備された，外部ストレージサブシステムを用いる場合がほとんどであり，その高い I/O スループットにより，共有ディスクアクセスが性能ボトルネックとなることを防いでいる．さらに，HDD に比べてランダムなリード I/O 性能が飛躍的に高い近年のフラッシュメモリ SSD を使えば，ディスク共用クラスタの性能ボトルネックを，大容量キャッシュを搭載したディスクコントローラを用いるよりも低コストで緩和させることが期待できる．

本報告は，メインフレーム製品での実装例を基に，ディスク共用 DB クラスタシステムにおける様々な設計上の課題を議論し，今後の分散システム構築の参考となることを最大の目的としている．システム設計上の課題は多岐に渡り，全てを詳しく網羅することはできないが，ここでは筆者が中心となって設計開発した，通常実行時の通信オーバヘッド削減に重点を置いた分散ロック管理方式を中心に議論を展開する．

2. 関 連 研 究

各種クラスタシステムの概観は Pfister[3] の，OLTP システム全般については Gary[4] らの教科書が参考になる．DB クラスタ構成形態の優劣については，古くは Stonebraker[5] が，非共用クラスタの優位性を主張している．しかし現時点では，OLTP 分野に限定すれば，ディスク共用クラスタの方が優勢である．ディスク共用クラスタと非共用クラスタには，それぞれ一長一短があり，最近ではどちらかを基本としながらも他方の要素を一部取り入れたハイブリッド型に進化する傾向も見える．

DBMS におけるロック制御については，Weikum ら[6] の教科書が包括的である．ディスク共用クラスタでのロック管理に関しては Mohan ら[7] による議論があり，実験システムでの実装については Bernstein[8] らの報告が参考になる．複数ノード間でロックマスタを合議決定するコンセンサス制御に関しては，Lamport の Paxos プロトコル[9] が基本である．

ディスク共用 DB クラスタについては，アカデミックな世界での議論よりも各ベンダによる製品提供の方が先行している[⋆1]．網羅的ではないが，現在稼動中のシステムで利用されている製品例をいくつか挙げると，まずメインフレームでは，クラスタ制御専用ノード（Coupling Facility）を設ける IBM の Parallel Sysplex[10] が代表的である．このシステムにおけるロックマネジャについては，Obermarck ら[11] の公開情報がある．ほぼ同じ仕組みのオープンシステム版（DB2 pureSclae[12]）も発表されている．IBM メインフレームでは，これとは別にディスクコントローラ装置にロック機能を持たせる Transaction Processing Facility (TPF)[13] の方式もある．オープンシステム系の代表例は，Oracle Real Application Cluster(RAC)[14] である．

国内の IT ベンダも，メインフレーム向けのディスク共用クラスタシステムを各々発表している．もともとは，都市銀行の金融第 3 次オンラインシステム開発に向けて，高性能・高可用性なシステムへの要望が高まったことに応えたものである．富士通はノード間の多数決によってロック制御を行う方式（AIM/SRCF: Shared Resource Control Facility[15][16]）を，日本電気は専用のフォールトトレラントなロック制御プロセッサを用いる方式（ACOS MSCP: Multi System Control Processor[17][18]）を，日立製作所は分割したリソースごとにロックマスタを設ける方式（VOS3 ARF: Advanced Reliability Feature[1][19]）を，それぞれ提案している．本報告は，この日立製作所の製品のロックマネジャ部分[20] を取り上げたものである．

これらの製品群のロックマネジャをおおまかに分類すると，表 1 のようになる．縦軸は，専用ハードウエア（物理・仮想をともに含む）もしくは専用ロックノードを必要とするかどうか，横軸はロック処理が集中型か分散型かで区分している．

DBMS 関連以外では，クラスタ構成を採用する分散ファイルシステムのロック管理として，Google の Chubby[21] が報告されている．ただし，これらの分散システムが提供するロック機能は，ファイルロック相当の低頻度な利用環境を前提としており，OLTP 環境下でのトランザクショナルな DB ロックのように細粒度で高頻度に発生するロック要求を想定したものではない．

表 1 ディスク共用 DB クラスタ用ロックマネジャ方式の分類

	集中ロックマネジャ	分散ロックマネジャ
専用ハードまたは専用ノードあり	IBM Parallel Sysplex IBM pureScale 日本電気 MSCP	IBM TPF (MPLF)
専用ハードおよび専用ノードなし	−	Oracle RAC 富士通 SRCF 日立 ARF

⋆1 以下に記述する各ベンダの製品名に関する商標表示をここにまとめて掲示しておく；IBM, Parallel Sysplex, DB2, および pureScale は，International Business Machines Corporation の商標です．Oracle と Java は，Oracle Corporation およびその子会社，関連会社の米国およびその他の国における登録商標です．その他記載の会社名，製品名は，それぞれの会社の商標もしくは登録商標です．

Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

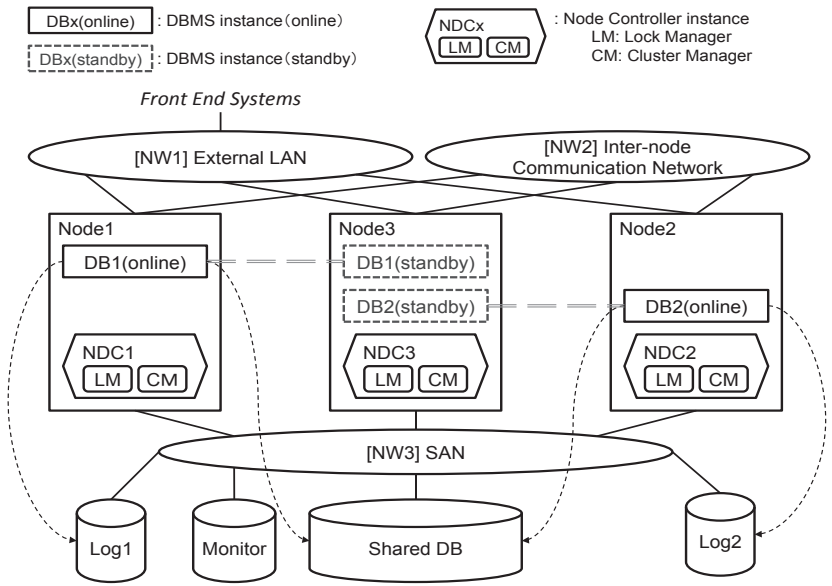

図 1 DB 共有クラスタシステム構成（3 ノード）

3. システムの構成と求められる要件

3.1 ディスク共用 DB クラスタシステムの構成

図 1 に，3 台の計算機ノードからなるディスク共有 DB クラスタシステムの構成例を示す．システムは，以下のような構成要素を含む[⋆1]；

計算機（ノード）： DBMS とノードコントローラが稼動するサーバ計算機であり，後述する 3 つのネットワークに接続される．各ノードには，クラスタを構成する最大ノード数を N として，$[0, N-1]$ の一意のノード番号が割り当てられる．

DBMS インスタンス（現用系/待機系）： 一連のリソースを共有して自己完結した DBMS 機能を実現するプロセス群．同じノード上で複数の DBMS インスタンスを動かしてもよい．また，現用系 DBMS インスタンスに対して待機系インスタンスを設け，現用系がダウンした場合に即座に処理を引き継がせることも可能である（ホットスタンバイ）．

ノードコントローラインスタンス： 各ノード上で動作するノード制御のプログラム．ノードコントローラは，内部に以下のようなサブコンポーネントを持つ；

- クラスタマネジャ：クラスタを構成するノードとその上で動く DBMS の状態を管理する．ハートビート監視を行って，ノードや DBMS インスタンスの起動，停止のイベントを検知してロックマネジャ等のコンポーネントに通知する．
- ロックマネジャ：共有 DB に対するロックを管理する．ロックマネジャは，リソースの論理名称だけを意識するため，どのようなリソース単位で排他制御が実行されるかは DBMS 次第である．多くの場合，DB ページがロック制御の単位となる．
- その他：本報告の議論上は付随的であるため図 1 には示していないが，ノードコントローラは，他にコミュニケーションマネジャとキャッシュマネジャを含む．

共有ストレージボリューム： 複数のノードから共有してアクセスされるストレージボリュームであり，以下の 2 種類からなる；

- DB ボリューム：複数の DBMS インスタンスから共用してアクセスされる DB を格納したストレージボリューム．全てのストレージボリュームは何らかの RAID レベルで保護されており，HDD の単点障害に対して耐久性を備える．
- モニタボリューム：クラスタを管理するために，ノードコントローラ間で共用されるストレージボリューム．複数ノードコントローラ間からのアクセス競合を逐次化するため，ボリュームレベルでの reserve/release を利用する．

占有ストレージボリューム： 特定のノードや DBMS インスタンスが占有してアクセスするストレージボリューム．OS が使うシステムボリュームや DBMS が使うログボリュームなどがある．煩雑化を避けるため，図 1 にはログボリュームだけを示している．

ネットワーク： 以下の 3 種類のネットワークを使う．性能を確保するためにはそれぞれ独立したネットワークとすることが望ましいが，縮退した簡易構成も可能である．

- 外部接続ネットワーク (NW1)：DB クラスタとそれを利用するフロントエンド側のシステムを接続するネットワーク．システム管理用のネットワークも含む．典型的にはイーサネットを L1/L2 層とする TCP/IP ネットワークである．
- ノード間通信ネットワーク (NW2)：DB クラスタ内のノードコントローラ間を接続するネットワーク．システム性能に大きく影響するので，専用の高速ネットワーク（InfiniBand など）を用いることが望ましい．

⋆1 メインフレーム製品版では，主要な構成要素の上限数は，ノード数 ≤ 8，DBMS インスタンス数 ≤ 32（待機系含まず）となっているが，この数字は便宜的なもので本質的な意味はない．

Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

- ストレージネットワーク (NW3)： DB クラスタの各ノードとストレージサブシステムを接続するネットワーク．典型的には FC-SAN を用いる．

3.2 システム設計上の要件

ディスク共用 DB クラスタシステムを設計・実装するに当たって前提となった要件と，それから導かれるいくつかの設計上の判断を述べる．

(1) 特別な新規ハードウエアもしくはクラスタ制御専用ノードを使用しない

開発コストと期間の制限から生じた要件である．状況によっては，クラスタ制御のための新規ハードウエアを設けて得られる効果（主として性能向上）が不利益を上回るという判断もありうるが，我々の場合は，この条件はプロジェクト初期から与えられた要件であった．

(2) 既存の DBMS を最小限の変更で利用可能とする

既にシングルノードで動く DBMS 製品が存在し，それをクラスタ対応に拡張することが目的であるため，この要件が必要となる．これは，我々に特有の状況というよりは，かなり一般的なものと想定してよいだろう．

(3) クラスタ化による性能劣化を最小限にとどめる

クラスタシステムでは，ノード間通信オーバヘッドのシステム性能に対する影響を必要最低限に抑えることが重要である．我々の場合，短いトランザクションを高トラフィックで処理するシステム，特に金融機関の勘定系を主な適用先としており，設計上の様々な選択もそれに対応したものとなっている．具体的な性能目標は，8 ノードまでスループットが線形に増大することと，1 トランザクションのレイテンシの悪化を 10%以下にすることである．

4. 分散ロック管理の設計と実装

ディスク共用 DB クラスタでのロック処理方式の基本は，ロック対象となるリソースを複数グループに分割し，グループ単位にロックマスタノードを設けるというものである[⋆1]．

4.1 リソースの分割とロックマスタの割当

ロックマネジャは，DBMS が発行するロック要求で与えられるリソース名称（の一部）をキーとしてリソースの属するグループを決定する．リソース名称からグループへのマッピング方法には何を用いてもよいが，我々の基本設計では，ハッシュまたはキーレンジ分割を使うようにしている．どちらの場合も，指定されたノードが停止中の場合は，やはり定義によって指定される，当該ノードのバックアップノードがマスタとなる．あるノードに対するバックアップノードは，優先順位付きで複数指定することができる．デフォルトでは，ノード番号 n のノードのバックアップノードは，$n+1$,$n+2$,$\cdots mod N$ である．

製品版では，ハッシュ分割機能は実装されておらず，キーレンジ分割だけが利用可能である．これは，障害時のシステムの振る舞いの予測可能性を重視したためである．主要な適用先である銀行勘定系の口座 DB を例に考えてみよう．クラスタシステムは高可用に設計されてはいるが，現実のシステム運用中には何らかの障害によってあるリソースグループのロックマスタが喪失した状態になることが想定される．これは，支店番号によるキーレンジ分割では，特定の支店群（例えば関東地方）の口座が利用不可能になることを意味する．一方，口座番号によるランダムなハッシュ分割では，全国の支店に渡って不特定多数の口座が利用不可能になってしまう．業務運用観点からは前者の方が望ましい．

表 2 ロック情報のレベル

レベル	ロック保有者の管理粒度	ロック対象リソースの管理粒度	情報保管場所
1	DBMS インスタンス	リソースグループ	メモリとモニタボリューム
2	DBMS インスタンス	ハッシュビットマップ	メモリとモニタボリューム
3	トランザクション	個別のリソース	メモリ

4.2 管理するロック情報の粒度（レベル）

ロックマネジャは，ロック情報を表 2 に示す 3 段階の粒度で管理する．各レベルのロック情報の使い方に関しては，後で詳しく述べる．

レベル 3 ロック情報は，個々のトランザクションが個々の共有リソースに対して保有する最も詳細なロック情報のことであり，以下の 3 つの状態を持つ；

保有： あるトランザクションが当該資源に対してロックを保有していることを示す．保有ロックモードには；EX(EXclusive), PU(Protected Update), PR(Protected Retrieval), SU(Shared Update), SR(Shared Retrieval) の 5 種類がある．

待ち： トランザクションが当該資源に対して出したロック要求が先行する保有ロックと衝突して認められず，先行ロックの開放を待っていることを示す．

保留： あるトランザクションが保有しているロックが，何らかの障害によってしばらくの間開放される見込みがない状態であることを示す．ロックマネジャは，当該資源に対する待ち状態のロック要求と新たに到着する全てのロック要求をエラーリターンする．

レベル 2 ロック情報は，レベル 3 情報を固定長（デフォルトでは 8,192 ビット）のビットマップに集約したものである．ビットマップは，DBMS インスタンス×リソースグループ

⋆1 全てのリソースグループのロックマスタを同一ノードにマッピングすれば，実質的に集中ロックマネジャと同じ構成になる．ただし，このような使い方をしている実例はない．

Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

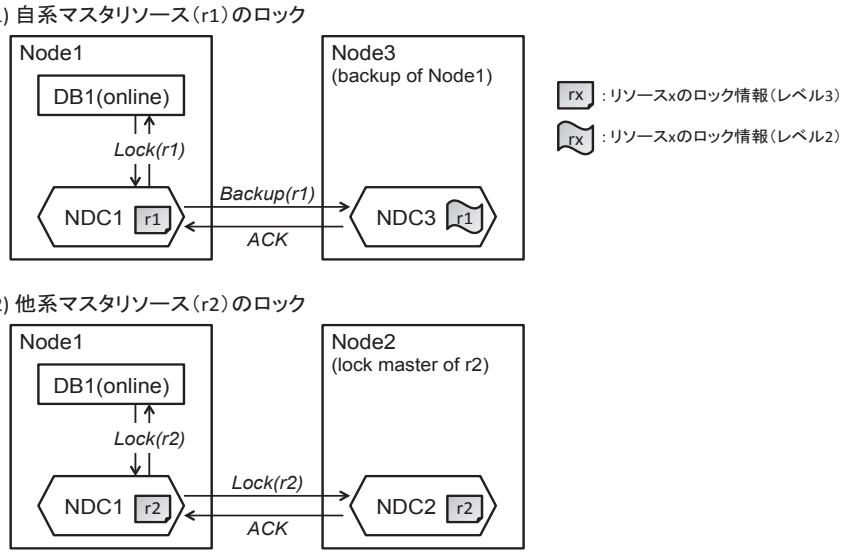

図 2 ロック処理概要（自系マスタおよび他系マスタ資源）

ごとに管理される．各ビット位置は，リソース名称をキーとしたハッシュ値に対応し，当該 DBMS インスタンス下で実行される単独または複数のトランザクションがそのリソースに対して EX モードのロックを保持している場合にオンとなる．

レベル 1 情報は，最も粗い情報であり，各 DBMS インスタンスがどのリソースグループをアクセスする可能性があるかを示す．レベル 2 とレベル 1 のロック情報は，保留状態を表すためだけに用いられる．

4.3 通常のトランザクションでのロック処理

図 2 に，通常のトランザクションで行われるロック処理概要を示す．ここで考慮すべきは，単一ノード障害に対応するため，複数ノード上にロック情報を冗長保持する方法である．

4.3.1 自系がマスタのリソースに対するロック： 図 2 の 1)

DB1 から，ノード 1 がロックマスタであるリソース $r1$ に対するロック要求が来た場合，NDC1 は自分でロック許可を判定することができる．ただし，この段階ではノード 1 上にしかロック情報が保持されていないため，自系のバックアップであるノード 3 にロック情報のコピーを送付する．これで，ロック情報はマスタノード（NDC1）とバックアップノード（NDC3）に 2 重化されることになる．

このバックアップノードへのロック情報コピー送付の通信オーバヘッドを削減するため，いくつかの工夫をしている．DB1 が DB ページ ($r1$) の更新を永続化する以前であれば，ノード 1 がダウンしても $r1$ のロックを保持し続ける必要はない[⋆1]．これを利用して，バックアップ情報の送付を個々のロック要求時に同期的に行うのではなく，DBMS のページ変更永続化時点（短いトランザクションであればこれは通常コミット時点になる）に一括して送付することで，メッセージ数を削減できる．また，個々の詳細なロック情報（レベル 3）をバックアップするのではなく，ハッシュビットマップ（レベル 2 ロック情報）を作ってその変更部分だけを送付することでメッセージサイズの削減と処理簡略化を実現している．

4.3.2 他系がマスタのリソースに対するロック： 図 2 の 2)

ノード 1 で稼働中の DB1 から，ノード 2 がロックマスタであるリソース $r2$ に対するロック要求が来た場合，NDC1 ロックマネジャは，リソース名称からロックマスタを決定して NDC2 ロックマネジャにロック要求を転送する．NDC2 はロック要求が許可可能か判定して NDC1 に応答を返す．許可可能でない場合は，このロック要求は NDC2 の内部で待ち状態に入る．ロックが許可された場合，そのロック情報（レベル 3）はロック要求を中継した NDC1 と，ロックマスタである NDC2 の 2 カ所に保持されることになる．

4.3.3 非トランザクションロック

上記のようなトランザクションに関連するロックの他に，ロックマネジャは「非トランザクションロック」も扱うことができる．ACID 属性とは関係しないが，実行プロセス間で排他制御が必要な資源（通信セッションなど）に用いられる．保有元がダウンした場合に保留されるのではなく開放されるという点がトランザクショナルなロックと異なる．

4.3.4 デッドロックの検出

デッドロックは，1 ノード内に閉じるもの（ローカルデッドロック）は待ちループの検出で，複数ノードにまたがるもの（グローバルデッドロック）はタイムアウトで検出する．

4.4 ロックマスタ交代

マスタ交代プロトコルは，マスタ不定状態から新たにマスタを設定する場合と，既にあるノードに設定されているマスタを別ノードに移動する場合の双方を扱う．

4.4.1 ロックマスタに関する状態とその遷移

ロックマネジャが管理するリソースグループの状態遷移を図 3 に示す．不活性 ($S0$) は初

⋆1 ACID 属性を備えた DB 更新における WAL(Write Ahead Log) プロトコルと同じ議論が適用できる．

Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

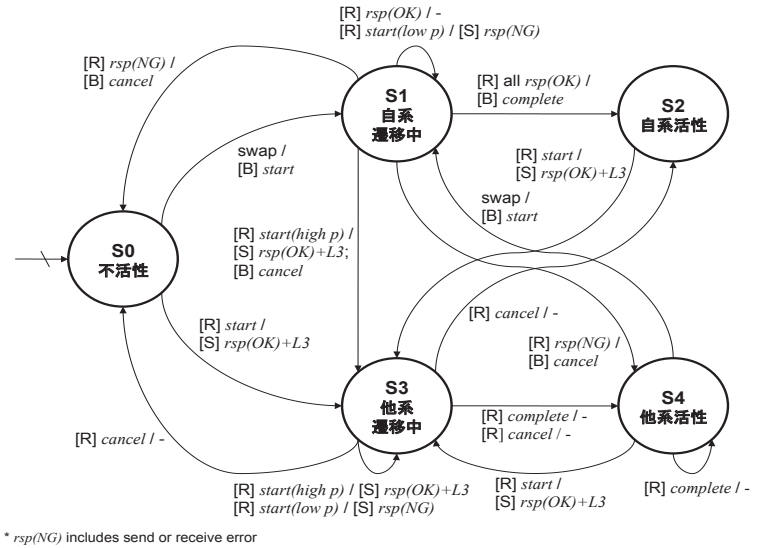

図 3 ロックマネジャの管理する状態遷移

期状態であり，ロック要求は拒絶される．定常状態では，リソースグループは自系がマスタである ($S2$) か，他系がマスタであるか ($S4$) のいずれかであり，この状態のリソースに対しては，4.3 で述べたロック処理が可能となる．$S1$ と $S3$ は，マスタ交代処理中の一時的な中間状態である．状態遷移図中の [S] はプロトコルメッセージの送信（シングルキャスト），[B] は複数ノードへの一斉送信，[R] は他ノードからのメッセージ受信を示す．

4.4.2 マスタ交代プロトコル

図 4 にロックマスタの交代処理プロトコルを示す．マスタ交代が発生する契機には，ノード起動，ノード停止（正常終了），ノードダウン（異常終了），オペレータコマンド（手動交代）などがある．マスタ交代プロトコルは，交代先ノードによって”pull”型で起動・推進される．したがって，交代元が不定の場合（$S0 \rightarrow S2$ の遷移）や，ノードダウンによる交代（$S4 \rightarrow S2$ の遷移）の場合にも対応できる．

プロトコルは，全ノードに”swap start”メッセージを一斉送信して交代の意図を伝え，全ノードから”OK”の応答を受領後”swap end”メッセージを一斉送信して新マスタ設定を通

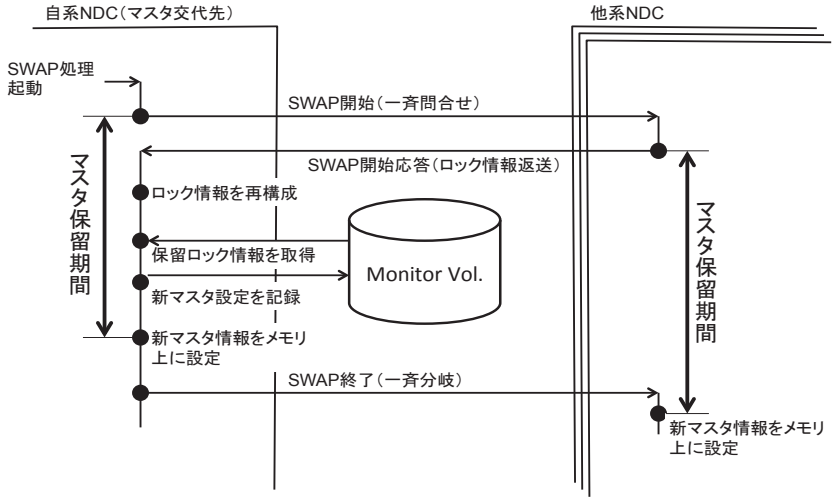

図 4 ロックマスタ交代処理概要

知するという 2 フェーズのメッセージ交換からなる．1 ノードでも”NG”応答をすれば，交代は失敗し，マスタは移動しない．新マスタの情報は，各ノードのノードコントローラ上のメモリ上だけでなく，共用のモニタボリュームにも書き込まれる．この書き込みは，モニタボリュームへの reserve/release によって逐次化される．図 4 の「マスタ保留期間」は，当該リソースグループがマスタ交代中の一時的な中間状態である期間を示す．

一般に分散処理系で各ノードの合意を得ようとする場合，Paxos に代表されるコンセンサスプロトコルを用いる必要がある．コンセンサスプロトコルは，各ノードがネットワークを介してのみ通信することと，各ノードがローカルな 2 次記憶装置だけを持つことを前提としている．我々のマスタ交代プロトコルは，純粋な分散処理プロトコルではなく，ネットワークによるメッセージ交換に加えて，ノード間共用ボリューム（モニタボリューム）へのアクセスを利用する．これにより，メッセージ交換部分の処理が簡略化できる．DB ボリュームの他に若干の管理用ボリュームを共有することは十分許容できる．

4.4.3 ロック情報の再構成

ロックマスタの交代では，交代先は交代元が保持していたのと同じロック情報を引き継ぐ必要がある．必要なロック情報は，クラスタ内の各ノードに分散して保持されているため，

Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

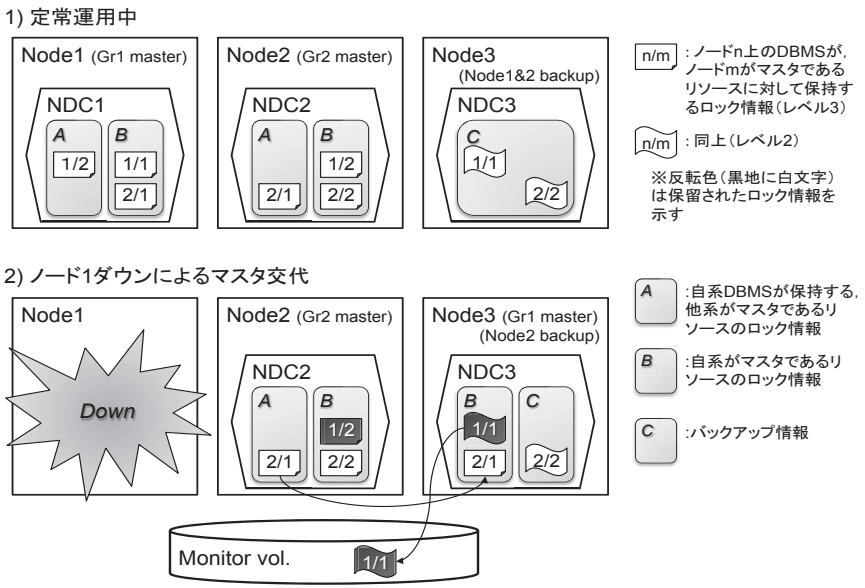

図 5 ロック情報の収集の仕方

交代先マスタはマスタ交代プロトコルのメッセージ交換を利用してそれを収集する．

交代元マスタが生きている場合，最も直接的で簡単な方法は，交代元マスタからの”swap start”応答メッセージに必要な情報を乗せることである．しかし，この方法はマスタノードダウン時の緊急のマスタ交代の場合は使えない．通常時（交代元マスタが生きている）と，緊急時（交代元マスタがダウンしている）で 2 種類の情報収集方式を使い分けることは，システムの複雑さを増すので望ましくない[⋆1]．このため，各ノードのロックマネジャが管理する，自系で稼働する DBMS インスタンスが保持するロックの情報を収集する方法を採用した．この方式では，通常時も緊急時も同じプロトコルが適用できる．

図 5 は，図 1 と同じ構成下でのロック情報管理の様子を詳細に示したものである．各ノード上のロックマネジャは；(A) 自系で動作する DBMS が保有する，他系がマスタであるリソースグループに関するロック情報，(B) 自系がロックマスタであるリソースグループに対して，クラスタ上の全 DBMS が保有するロック情報，(C) 自系がバックアップしている他系ノードコントローラが保有する，その系がマスタであるリソースグループに対してその系で動作する DBMS が保有するロック情報，の 3 種類のロック情報を保持する．例えば，図 5 の 1) において，NDC1 は，DB1 の全ロック情報（1/1 と 1/2）と，DB2 がリソースグループ 1 に対して保持するロック情報（2/1）を管理している．3 種類のロック情報のうち，(A)(B) はレベル 3 ロック情報であるのに対し，(C) はレベル 2 ロック情報である．

ここで，ノード 1 がダウンしたとしよう．リソースグループ 1 のマスタは，NDC1 のバックアップとして指定されている NDC3 に移動する必要がある．ノード 1 のダウン検出をクラスタマネジャから通知された NDC3 は，リソースグループ 1 に対するマスタ交代プロトコルを始動する．”swap start”メッセージの応答で，NDC2 は，DB2 がリソースグループ 1 に対して保有するレベル 3 のロック情報（2/1）を NDC3 に送る．NDC3 は，このレベル 3 情報（2/1）と，自分が保持している DB1 がリソースグループ 1 に対して保持しているロックのレベル 2 情報（1/1）を用いて，リソースグループ 1 に対してクラスタ上の全 DBMS(DB1 と DB2) が保持しているロックの情報を再構成して新たなマスタとなる．DB2 のロック情報は全てレベル 3 であるため，DB2 は通常のトランザクション処理を継続することができる．一方，ダウンした DB1 のレベル 2 ロック情報は，保留状態を表し，該当する資源に新たなロック要求がきた場合にエラーリターンするために使われる．

レベル 2 のロック情報は，資源名をキーとしたハッシュによって集約されたものであり，ハッシュの衝突によって本来拒否されるべきではないロック要求がエラーリターンする可能性がある．この保留ビットマップは，DB1 の待機系，またはいずれかのノードで再起動された DB1 インスタンスが，ダウン時に仕掛り中のトランザクションの redo/undo 回復処理を完了した時点でクリアされる．待機系がある場合，ダウン発生から回復処理完了までにかかる時間はおおよそ 10^1sec である．したがって，レベル 2 ロック情報による保留状態は，システム運用上問題となるほど長く続かないことが期待できる．

DB1 の回復処理が終了しないうちに全ノードが停止すると，メモリ上に保持される 1/1 のレベル 2 ロック情報が失われてしまうため，NDC3 はこの情報をモニタボリュームにも記録する．何らかの障害によって，レベル 3 もレベル 2 の情報も得られなかった場合，NDC3 はモニタボリュームに記録されているレベル 1 の情報を使って，DB1 がアクセスしていた可能性のあるリソースグループ 1 に対する新たなロック要求を拒否する．このように，新たなマスタは，3 段階の粗さのロック情報を用いて，ロック管理を継続する．ロックマスタの決定時には，モニタボリュームへのアクセスが必須であり，ノード間通信ネットワークの分

⋆1 KISS (Keep It Simple, Stupid!) は，OLTP システム設計者が遵守すべき黄金律である．

Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

断時でもマスタが 2 重に設定されたり，マスタが失われたりする危険はない．

4.4.4 マスタ交代に関する補足

(1) 複数のノードによるマスタ交代プロトコル処理が競合した場合は，ノード番号の小さい方の処理を優先させる．ノードコントローラは，競合する”swap start”を受信した場合，交代先ノードのノード番号を比較して，交代ノードの番号が小さい方には”OK”を，大きな方には”NG”を応答する．

(2) 複数のイベントがほぼ同時に発生した場合，各ノードコントローラでのイベントの受領と処理順序によって，マスタが複数ノード間を頻繁に移動することがある．クラスタシステムの完全停止状態から全ノードを一斉に立ち上げる場合がその典型的な例である．この問題を軽減するため，マスタ交代処理を一定時間後にリトライする場合，ノード番号に対応して単調増加する値をリトライ間隔として，ノード番号が小さい処理を優先させる．

4.5 ノード間通信障害への対応

図 2 の 2) において，ノード 1 とノード 2 の間の通信路が障害になったとしよう．NDC1 は DB1 からのリソースグループ 2 に対する新たなロック要求を拒否するが，保有済みのロックに対するアンロック要求は受け付ける．他方 NDC2 は，DB1 がリソースグループ 2 に対して保有する全てのロックを保留状態に移行する．このため，NDC1 と NDC2 が保有する DB1 のレベル 3 ロック情報に差分が生じる．この差分は，ノード間の通信障害が回復した時点で，NDC1 が NDC2 に DB1 の実行中トランザクションのリストを送付することで解消される．NDC2 は，送られてきたリストと自分で管理している DB1 上のトランザクション情報を比較し，既に終了したトランザクションが保有するロックを全て解放する．

図 2 の 1) で，ノード 1 とノード 3 の通信路が障害になった場合は，NDC3 によるバックアップは中止され，別の新たなバックアップノード（この場合は NDC2）が設定される．ノード 1 とノード 3 の間の通信が回復した時点で，改めて NDC1 は NDC3 とバックアップペアを組み，NDC2 によるバックアップは解消される．バックアップノードの再設定の場合は，差分ではなく全ビットマップ情報を改めて送る．

ノードダウンとノード正常終了で同じマスタ交代処理を適用したのと同様，ノード正常開始とノード間通信回復でも処理を共用している．これにより，イベント種別ごとに対応処理が細分化されることを防ぎ，不良を作りこむ可能性を低減させている．

4.6 その他の関連する処理

ディスク共用 DB クラスタシステムを構築するために必要ではあるが，本報告では触れない機能コンポーネントには以下のようなものがある；

モニタボリューム管理： 共用のモニタボリュームを管理する

ログマージユティリティ 各 DBMS がローカルに出力するログファイルをマージする

分散キャッシュ管理： 各 DBMS が管理するメモリ上のキャッシュの整合性を管理する

5. 評　　価

5.1 設計上の要求に対する充足度

これまで述べてきたディスク共用クラスタ方式が，3.2 に示した設計上の要件をがどのように満たしているかを評価する；

(1) 特別な新規ハードウエアもしくはクラスタ制御専用ノードを使用しない

ノードコントローラは，DBMS と同じ計算機ノード上で動くユーザモードのソフトウエアとして実装されており，ロック専用のハードウエアおよびロック専用ノードを必要としない．このため，オープン環境への移植も容易である．

(2) 既存の DBMS を最小限の変更で利用可能とする

クラスタ構成に対応するために DBMS 側に必要な変更は以下の 3 点である；

- DBMS 内部のロックマネジャに，ノードコントローラへロック要求を中継するゲートウエイ機能を実装すること
- （必要であれば）ノードコントローラのキャッシュマネジャ機能と連携した DB キャッシュ制御処理を実装すること
- ログマージのためのタイムスタンプ情報をログヘッダに記録できるようログフォーマットと処理を変えること

実際にどの程度の変更が必要かは個々の DBMS に依存するが，我々のメインフレーム版製品では，開発プロジェクト上許容可能な範囲に収まった．

(3) クラスタ化による性能劣化を最小限にとどめる

具体的には，8 ノードまでのスループットの線形増加と，1 トランザクションあたりのレイテンシ悪化を 10%以下に抑えることである．我々の分散ロック処理方式のオーバヘッドはノード数に依存しないため，スループット増加が実現可能であるかどうかは，共用 DB ボリュームへの I/O 性能限界にほぼ依存する．金融システムへの応用を想定して，ここでは TPC-A[22] を評価用の処理モデルとして採用する[⋆1]．現実的なシステムでは，TPC-A を構

⋆1 性能ベンチマークとしては 1995 年に廃止されたが，オープンに利用できる「単純で短い」業務処理モデルとしてはまだ有効性が残っている．

Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

成する DB のうち TELLER と BRANCH はローカルなインメモリ処理となるため I/O は発生しない．また，HISTORY レコードは ACCOUNT レコードに包含される（階層データモデルを想定）．結局，1 トランザクションで発生する共用 DB ボリュームへの I/O は；a) ACCOUNT レコードに対する B-tree インデックスのリーフノードのリード 1 回（上位のインデックスはメモリ常駐），b) ACCOUNT レコードのリードとライト各 1 回，の計 3 回である．ノードあたり 1000tps と仮定すると，8 ノードで 8000tps（大手金融機関でも十分な性能）であり，I/O 密度は $3 * 8000 = 24kIOPS$ となる．この程度の I/O 負荷は，エンタープライズクラスの外部 RAID 装置を使えば問題なく対応できる．

トランザクションのレイテンシの悪化は，分散ロック処理でのノード間通信により引き起こされる．インデックスの下位 2 層のレコード（それより上位のインデックスには変更が発生しないようシステム的に運用）と ACCOUNT レコードに対してロックが必要であるとして，1 トランザクションでは，個別ロック要求 3 回＋全アンロック要求 1 回の 4 回のロック処理が発生する．リモートノードがマスタの場合は通信が 4 回 (ノード間での 1 往復のメッセージのやり取りを通信 1 回と数える)，ローカルノードがマスタの場合は通信が 2 回（トランザクションコミット時にバックアップ情報をビットマップでまとめて送信）となるため，ローカル処理比率を ρ（$0 \leq \rho \leq 1$）とすると通信回数の期待値は $2\rho + 4(1-\rho) = 4 - 2\rho$ となる．1 回の通信単価を 1ms と仮定すると，分散ロック処理による遅延は全体で 2〜4ms となる．1 トランザクションは，端末通信を含めて 100ms 程度かかるとすると，ロックによる通信オーバヘッドは許容範囲に収まる．また，アンロック要求の通信を次のトランザクションのロック要求とまとめたり，複数トランザクションのロック要求をまとめたりすることで通信回数を実質的に減らす余地も残されている．

5.2 集中ロックマネジャとの比較

2 重化された集中型のロックマネジャでは，トランザクションあたりのロック回数 n に対して $2(n+1)$ 回の通信が固定的に発生する．これは，我々の方式の 2 倍以上である．具体的な性能差は，ノード間通信方式の実装に依存する．

ロックマネジャノードがダウンした時，集中型では 2 重化された交代ノードに全てのロック情報がコピーされているため，継続処理は単純である．我々の方式では，クラスタ内に分散されたロック情報をマスタ交代プロトコルを用いて収集する必要があり，オーバヘッドが大きくなる．我々の方式は，通常のロック処理オーバヘッドを抑えることに重点を置き，その分のしわ寄せをマスタ交代処理に負わせていると言えよう．しかし，この収集処理はほとんどの場合 10^1〜10^2ms 程度で完了するので，システム的に問題となることはない．

5.3 その他： 信頼性，拡張・保守性など

Gray らは文献 4) の p485 で，”Don’t get stuck maintaining a lock manager, and if you do, take out all the optimizations and fancy features as a first step.” と忠告している．シングルノードの DBMS においても，開発量は小さい（〜10KLOC）がタイミングバグを根絶させるのは困難なロックマネジャを，クラスタ環境で分散協調させるのは無謀な挑戦であるとも言えよう．この観点からすると，集中型ロックマネジャの方が単純で望ましい．

6. まとめと今後の方針

ディスク共用 DB クラスタシステムにおける分散ロックマネジャの処理方式を示した．定常状態でのトランザクション処理オーバヘッドを小さくすることを優先し，その代償としてノードダウンに伴うロックマネジャの交代時にクラスタ内に分散して保持されたロック情報を収集する手間をかける仕組みを採用している．集中型ロックマネジャと比較して内部処理が複雑になるが，注意深く設計・実装・テストすれば実用上問題ない品質を確保することが可能である．DBMS と同じノード上で稼動するユーザモードのプログラムとして実装されており，特別なハードウエアやロックマネジャ専用ノードを設けずにシステムを構成できる．メインフレーム版の製品は金融機関など大規模 OLTP システムで利用されている．

今後の発展方向としては，SSD 共用 DB クラスタ構成に向けた方式最適化がありうる．分析応用主体のストレージ非共用クラスタ技術と OLTP 用途主体のストレージ共用クラスタ技術は，今後も並列して発展してゆき，製品レベルでは両者の技術を取り入れたハイブリッド型も提供されてゆくであろう．

参 考 文 献

1) 日立製作所：VOS3 DB/DC 高信頼化機能 ARF シリーズ解説・文法・操作書 (1998). 資料番号: 6190-6-334.
2) Brewer, E.A.: Towards Robust Distributed Systems (2000). (Invited talk) Symposium on Principles of Distributed Computing.
3) Pfister, G.F.: *In Search of Clusters*, Prentice Hall, Upper Saddle River, 2nd edition (1998).
4) Gray, J. and Reuter, A.: *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, San Francisco (1993).
5) Stonebraker, M.: The Case for Shared Nothing, *IEEE Database Engineering*, Vol.9, No.1, pp.610–621 (1986).
6) Weikum, G. and Vossen, G.: *Transactional Information Systems*, Morgan Kauf-

Vol.2012-DBS-154 No.11
Vol.2012-IFAT-107 No.11
2012/8/2

mann, San Francisco (2002).
7) Mohan, C. and Narang, I.: Efficient Locking and Caching of Data in the Multisystem shared Disk Transaction Environment, *Proceedings of the 3rd International Conference on Extending Database Technology*, Lecture Notes in Computer Science, No.580, Springer-Verlag, Berlin, pp.453–468 (1992).
8) Bernstein, P.A., Shipman, D.W. and RothnieJr., J.B.: Concurrency Control in a System for Distributed Databases (SDD-1), *ACM Transactions on Database Systems*, Vol.5, No.1, pp.18–51 (1980).
9) Lamport, L.: The part-time parliament, *ACM Transactions on Computer Systems*, Vol.16, No.2, pp.133–169 (1998).
10) IBM: *IMS/ESA Data Sharing in a Parallel Sysplex* (1997). Form Number: SG24-4303-00.
11) Obermarck, R.L., Strickland, J.P. and Watts, V.L.: Method and Means for the Sharing of Data Resources in a Multiprocessing, Multiprogramming Environment (1983). U.S. Patent 4399504.
12) IBM: *Highly Available and Scalable Systems with IBM eX5 and DB2 pureScale* (2011). Redbooks: REDP-4742-00.
13) IBM: *IBM z/Transaction Processing Facility: Overview and Enterprise Integration using SOA* (2010). Redbooks: REDP-4611-00.
14) Oracle: *Oracle Real Application Clusters Administration and Deployment Guide 11g Release 2 (11.2)* (2011). Part Number: E16795-11.
15) IPSJ コンピュータ博物館：【富士通】OSIV/F4 MSP. http://museum.ipsj.or.jp/computer/os/fujitsu/0010.html.
16) 高崎喜久夫：共用資源の占有決定処理方式 (1983). 日本国特許 特開昭 58-172764.
17) IPSJ コンピュータ博物館：【日本電気】ACOS-6/MVX. http://museum.ipsj.or.jp/computer/os/nec/0015.html.
18) 野見山眞一：ファイル制御処理装置 (1984). 日本国特許 特開昭 59-81748.
19) 日立製作所：VOS3 TMS-4V/SP 解説 (2011). 資料番号: 6190-6-104-B1.
20) 新田　淳，鈴木恭一，住吉孝史：システム間データベース共用方式 (1987). 日本国特許 特開昭 62-145349.
21) Burrows, M.: The Chubby lock service for loosely-coupled distributed systems, *7th Symposium on Operating System Design and Implementation (OSDI '06)*, Seattle (2006).
22) Transaction Processing Performance Council: *TPC Benchmark A - Standard Specification, Revision 2.0* (1994). http://www.tpc.org/tpca/spec/tpca_current.pdf.