

Oracle8i Parallel Server

概要

リリース 8.1

2000年2月

部品番号 : J00956-01

ORACLE®

Oracle8i Parallel Server 概要, リリース 8.1

部品番号 : J00956-01

原本名 : Oracle8i Parallel Server Concepts, Release2 (8.1.6)

原本部品番号 : A76968-01

原本著者 : Mark Bauer

原本協力者 : Wilson Chan, Sohan Demel, Merrill Holt, Michael Zoll, Christina Anonuevo, Lance Ashdown, David Austin, Bill Bridge, Sandra Cheever, Carol Colrain, Mark Coyle, Connie Dialeris, Karl Dias, Anurag Gupta, Deepak Gupta, Mike Hartstein, Andrew Holdsworth, Ken Jacobs, Ashok Joshi, Jonathan Klein, Jan Klokkers, Boris Klots, Anjo Kolk, Tirthankar Lahiri, Bill Lee, Lefty Leverenz, Juan Loaiza, Sajjad Masud, Neil Macnaughton, Ravi Mirchandaney, Rita Moran, Kant Patel, Erik Peterson, Mark Porter, Darryl Presley, Brian Quigley, Ann Rhee, Pat Ritto, Roger Sanders, Hari Sankar, Ekrem Soylemez, Vinay Srihari, Bob Thome, Alex Tsukerman, Tak Wang, Graham Wood, Betty Wu

Copyright © 1996, 1999, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム (ソフトウェアおよびドキュメントを含む) の使用、複製または開示は、オラクル社との契約に記載された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation (米国オラクル) または日本オラクル株式会社 (日本オラクル) を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation (米国オラクル) およびその関連会社は一切責任を負いかねます。当プログラムを米国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに	ix
第1部 パラレル処理の基本	
1 Oracle Parallel Server の概要	
Oracle Parallel Server の概要	1-2
Oracle Parallel Server の効果	1-2
スケーラビリティ	1-3
高可用性	1-3
透過性	1-3
2 パラレル・ハードウェア・アーキテクチャ	
クラスタ・ハードウェア・コンポーネントの概要	2-2
ノードの概要	2-2
メモリー・アクセス	2-2
均一メモリー・アクセス	2-3
不均一メモリー・アクセス	2-3
高速インターコネクト	2-4
クラスタ/ノードおよびインターコネクト	2-5
クラスタ化システムでのストレージ・アクセス	2-5
均一ディスク・アクセス	2-5
不均一ディスク・アクセス	2-6
様々なクラスタ上で動作する Oracle Parallel Server	2-8

3 Oracle Parallel Server のアーキテクチャ

クラスタ化システムに対する Oracle Parallel Server コンポーネント	3-2
クラスタ化システムに対するコンポーネントの概要	3-2
Cluster Manager	3-3
分散ロック・マネージャ	3-4
クラスタ・インターコネクトおよびプロセス間通信（ノード間）	3-6
ディスク・サブシステム	3-6

第 II 部 Oracle Parallel Server のロック処理

4 インスタンス間の調整

同期化	4-2
ローカル・ロック	4-3
グローバル・ロック	4-4
非パラレル・キャッシュ管理の調整	4-5
非パラレル・キャッシュ管理ロック	4-5
非 PCM グローバル・ロック	4-5
非パラレル・キャッシュ管理ロックの概要	4-6
パラレル・キャッシュ管理の調整	4-8
パラレル・キャッシュ管理処理の例	4-10
ブロック・レベルのロック	4-13

5 パラレル・キャッシュ管理

パラレル・キャッシュ管理およびロック実装	5-2
キャッシュ一貫性矛盾の解決におけるキャッシュ・フュージョンの役割	5-2
ロック期間および細分性	5-4
2つのタイプのロック期間	5-4
2つのタイプのロックの細分性	5-5
ロックのコスト	5-6
分散ロック・マネージャによるロック・メカニズムの調整	5-7
リソース・アクセス権としてのロック・モード	5-8
インスタンスによる分散ロック・マネージャ・リソースへのデータベース・リソースのマップ	5-10
分散ロック・マネージャによるロック情報の記録	5-10
分散ロック・マネージャ・ロックとグローバル・ロックの関連	5-11

リソースにおけるインスタンスごとの1つのロック	5-13
ロック要素およびパラレル・キャッシュ管理ロック	5-14
固定パラレル・キャッシュ管理ロックのロック要素	5-14
解放可能パラレル・キャッシュ管理ロックのロック要素	5-15
1:1 パラレル・キャッシュ管理ロックのロック要素	5-15
パラレル・キャッシュ管理ロックの動作方法	5-16
インスタンス LCK プロセスによって所有されるパラレル・キャッシュ管理ロック	5-17
複数インスタンスによる同じロックの所有	5-17
1:1 ロックの動作方法	5-18
パラレル・キャッシュ管理ロックごとのブロック数	5-20
複数ブロックを処理するロックの例	5-21
固定パラレル・キャッシュ管理ロックの周期性	5-22
ping: 更新の必要性の通知	5-22
ロック・モードおよびバッファ状態	5-24
DLM によるリソース・ロック要求の許可および調整方法	5-26
ロックの割当ておよび期間の指定	5-31
パラレル・キャッシュ管理ロックごとのブロック数	5-31
ロックの細分性の選択	5-32
固定ロックおよび解放可能ロックの同時使用	5-33
グループ所有のロック	5-33
マルチスレッド・サーバーおよびXA に対する分散ロック・マネージャのサポート	5-33
分散ロック・マネージャに対するメモリー要件	5-34

第 III 部 Oracle Parallel Server の実装

6 Oracle Parallel Server コンポーネント

Oracle Parallel Server のインスタンスおよびデータベース・コンポーネント	6-2
Parallel Server 固有のプロセス	6-2
Oracle Parallel Server プロセスの概要	6-4
キャッシュ・フュージョン処理およびブロック・サーバー・プロセス	6-5
システム変更番号処理	6-6
Lamport SCN 生成の動作方法	6-6

7 Oracle Parallel Server の記憶域上の考慮事項

Oracle Parallel Server 固有の記憶域に関する問題点	7-2
--	-----

データ・ファイル	7-2
REDO ログ・ファイル	7-3
ロールバック・セグメント	7-6
領域管理および空きリスト・グループ	7-10
Oracle による空き領域の処理方法	7-10
セグメント、エクステントおよび最高水位標	7-10
空きリストおよび空きリスト・グループ	7-12
空きリスト・グループ	7-13
セグメント・ヘッダーおよび空きリストに対する競合回避	7-14
空きリスト・グループの例	7-16
空きリスト・グループを使用したデータのパーティション化	7-19
Oracle による空きリスト・グループのパーティション化方法	7-19
インスタンス、ユーザーおよびロックと空きリスト・グループの対応付け	7-20
インスタンスと空きリストの対応付け	7-20
ユーザー・プロセスと空きリストの対応付け	7-21
PCM ロックと空きリストの対応付け	7-21
空き領域管理のための SQL オプション	7-24
エクステント割当ての制御	7-25
新しいエクステントの自動割当て	7-25
新しいエクステントの事前割当て	7-25
セグメントの最高水位標の移動	7-26

8 スケーラビリティおよび Oracle Parallel Server

Oracle Parallel Server のスケーラビリティ機能	8-2
拡張されたスループット: スケールアップ	8-2
スピードアップおよびスケールアップ: パラレル処理の目標	8-3
パラレル処理が有利な場合	8-5
意思決定支援システム	8-5
異なるデータ・ブロックを更新するアプリケーション	8-6
部門別アプリケーション	8-6
アプリケーション・プロファイル	8-7
マルチノード・パラレル実行	8-8
クライアント対サーバー接続の概要	8-8
マルチスレッド・サーバーの使用によるスケーラビリティの拡張	8-8
複数リスナーに対する Connect-Time Failover	8-10

複数リスナーに対するクライアント・ロード・バランス	8-10
スケーラビリティの4つのレベル	8-10
ハードウェアおよびネットワークのスケーラビリティ	8-11
オペレーティング・システムのスケーラビリティ	8-11
データベース管理システムのスケーラビリティ	8-12
アプリケーションのスケーラビリティ	8-12
シーケンス・ジェネレータ	8-13
Oracle Parallel Server での Oracle パラレル実行	8-15

9 高可用性および Oracle Parallel Server

高可用性の概念	9-2
可用性の評価	9-2
高可用性の基準	9-2
停止の原因	9-3
高可用性の計画	9-4
システム・レベルの計画	9-4
Oracle Parallel Server および高可用性	9-5
クラスタ・コンポーネントおよび高可用性	9-5
災害時計画	9-7
障害保護妥当性チェック	9-7
フェイルオーバーおよび Oracle Parallel Server システム	9-8
フェイルオーバーの基本	9-8
フェイルオーバーの期間	9-9
クライアント・フェイルオーバー	9-9
透過的アプリケーション・フェイルオーバーの使用	9-10
サーバー・フェイルオーバー	9-12
ホストベース・フェイルオーバー	9-12
Oracle Parallel Server フェイルオーバー	9-13
Oracle Parallel Server のフェイルオーバーの動作方法	9-13
Oracle Parallel Server における高可用性の構成	9-17
デフォルト N ノード Parallel Server 構成	9-17
高可用性の基本構成	9-18
高可用性の共有ノード構成	9-24
高可用性の配置に向けて	9-25

第 IV 部 参照情報

A リリース間の相違点

リリース 8.1 とリリース 8.1.6 の相違点	A-2
新機能	A-2
廃止されたパラメータ	A-2
廃止された統計情報	A-2
新しい統計情報	A-2
デフォルトのパラメータ設定の変更	A-2
リリース 8.0.4 とリリース 8.1 の相違点	A-3
キャッシュ・フュージョン・アーキテクチャの変更	A-3
新しいビュー	A-3
GMS の削除	A-4
パラレル・トランザクション・リカバリから「ファースト・スタート・パラレル・ ロールバック」への改名	A-4
インスタンス登録への変更	A-4
リスナーのロード・バランス	A-5
診断プログラムの拡張	A-5
Oracle Parallel Server Management (OPSM)	A-5
Parallel Server のインストールおよびデータベース構成	A-5
ジョブとインスタンスの親和性	A-6
廃止されたパラメータ	A-6
リリース 8.0.3 とリリース 8.0.4 の相違点	A-7
新しい初期化パラメータ	A-7
廃止された初期化パラメータ	A-7
廃止された起動パラメータ	A-7
動的パフォーマンス・ビュー	A-7
グループ・メンバーシップ・サービス	A-7
リリース 7.3 とリリース 8.0.3 の相違点	A-8
新しい初期化パラメータ	A-8
廃止された GC_* パラメータ	A-8
変更された GC_* パラメータ	A-8
動的パフォーマンス・ビュー	A-9
グローバル動的パフォーマンス・ビュー	A-9
分散ロック・マネージャ	A-9

インスタンス・グループ	A-10
グループ・メンバーシップ・サービス	A-10
ファイングレイン・ロック	A-10
クライアント側のアプリケーション・フェイルオーバー	A-10
Recovery Manager	A-10
リリース 7.2 とリリース 7.3 の相違点	A-11
初期化パラメータ	A-11
データ・ディクショナリ・ビュー	A-11
動的パフォーマンス・ビュー	A-11
空きリスト・グループ	A-11
ファイングレイン・ロック	A-11
インスタンス登録	A-12
ソートの改善	A-12
XA パフォーマンスの向上	A-13
XA リカバリ拡張	A-13
遅延トランザクション・リカバリ	A-14
接続時のロード・バランス	A-15
ソート・オペレーションのキャッシュの回避	A-15
遅延ロギング・ブロック・クリーンアウト	A-16
パラレル実行プロセッサの親和性	A-17
リリース 7.1 とリリース 7.2 の相違点	A-17
領域の事前割当ては不要	A-17
データ・ディクショナリ・ビュー	A-17
動的パフォーマンス・ビュー	A-18
空きリスト・グループ	A-18
表ロック	A-18
ロック・プロセス	A-18
リリース 7.0 とリリース 7.1 の相違点	A-19
初期化パラメータ	A-19
動的パフォーマンス・ビュー	A-19
バージョン 6 とリリース 7.0 の相違点	A-19
バージョン互換性	A-19
ファイル操作	A-19
遅延ロールバック・セグメント	A-21
REDO ログ	A-21

空き領域リスト	A-22
SQL*DBA	A-22
初期化パラメータ	A-23
アーカイブ	A-23
メディア・リカバリ	A-24

B 制限事項

共有モードと排他モード間の互換性	B-2
エクスポート・ユーティリティおよびインポート・ユーティリティ	B-2
共有モードと排他モード間の互換性	B-2
制限事項	B-2
一度に割り当てられるブロックの最大数	B-2
共有モードでの制限事項	B-3

索引

はじめに

このマニュアルでは、パラレル処理を正常に実装するために必要な、Oracle Parallel Server の概念を説明します。このマニュアルの情報は、あらゆるオペレーティング・システム上で動作する Oracle Parallel Server に適用されます。

このマニュアルを読んでから、『Oracle8i Parallel Server セットアップおよび構成ガイド』および『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を読んでください。Oracle および Oracle Server の管理についての一般的な情報は、『Oracle8i 概要』および『Oracle8i 管理者ガイド』を参照してください。

Oracle8i での新規事項

このマニュアルは、Oracle8i 用に改訂されています。Oracle8i では、キャッシュ・フュージョンが導入されています。この機能は、インスタンス間の競合によって生じる、読み込み / 書き込みの競合を解消する際のオーバーヘッドを削減します。これによって、Oracle Parallel Server のスケーラビリティおよびパフォーマンスが大幅に向上します。

参照： Oracle Parallel Server のリリース間の機能変更については、[付録 A 「リリース間の相違点」](#) を参照してください。

リリース 8.1.5

リリース 8.1.5 では、キャッシュ・フュージョンの 1 次段階が導入されています。

リリース 8.1.6

リリース 8.1.6 では、プライマリ / セカンダリ・インスタンス機能およびキャッシュ・フュージョンへのさらなる拡張が導入されています。また、いくつかの新しいパフォーマンス統計情報もあります。

対象読者

このマニュアルは、Oracle Parallel Server を使用して作業を行うデータベース管理者およびアプリケーション開発者を対象としています。

このマニュアルの構成

このマニュアルでは、Oracle Parallel Server の概念を 4 部にわたって説明しています。まず、Oracle Parallel Server に対するパラレル処理の基本について説明します。次に、インスタンス間の同期化処理について説明し、Oracle Parallel Server の実装方法の基本について説明します。最後に参照情報として、バージョン間の相違点および Oracle Parallel Server の実装制限について説明します。

構成内容

このマニュアルは、次の 4 部で構成されています。

第 1 部「パラレル処理の基本」

第 1 章「Oracle Parallel Server の概要」

この章では、オンライン・トランザクション処理および意思決定支援アプリケーションで有効なパラレル処理およびパラレル・データベース・テクノロジーについて説明します。

第 2 章「パラレル・ハードウェア・アーキテクチャ」

この章では、クラスタ環境の特徴を表すハードウェア・コンポーネントおよび高レベルのアーキテクチャ・モデルについて説明します。

第 3 章「Oracle Parallel Server のアーキテクチャ」

この章では、単一インスタンス・コンポーネントに加えて、Parallel Server 処理用に提供されるアーキテクチャ・コンポーネントについて説明します。

第 II 部「Oracle Parallel Server のロック処理」

第 4 章「インスタンス間の調整」

この章では、クラスタに関するインスタンス間の調整アクティビティについて詳しく説明します。

第 5 章「パラレル・キャッシュ管理」

この章では、パラレル・キャッシュ管理のロックについて詳しく説明します。

第 III 部「Oracle Parallel Server の実装」

第 6 章「Oracle Parallel Server コンポーネント」

この章では、Oracle Parallel Server アプリケーションに対する実装コンポーネントについて説明します。

第 7 章「Oracle Parallel Server の記憶域上の考慮事項」

この章では、Oracle Parallel Server アプリケーションに対する記憶域上の考慮事項について説明します。

第 8 章「スケーラビリティおよび Oracle Parallel Server」

この章では、Oracle Parallel Server のスケーラビリティ機能について説明します。

第 9 章「高可用性および Oracle Parallel Server」

この章では、Oracle Parallel Server を使用して高可用性を実装する最も実質的な方法と、その概念について説明します。

第 IV 部「参照情報」

付録 A「リリース間の相違点」

この付録では、Oracle Parallel Server に関する Oracle の今回のリリースと以前のリリースとの相違点について説明します。

付録 B「制限事項」

この付録では、Oracle Parallel Server に対する制限事項を説明します。

関連ドキュメント

このマニュアルを読んだ後に、『Oracle8i Parallel Server セットアップおよび構成ガイド』および『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を読んでください。

詳細は、次のマニュアルを参照してください。

インストール・ガイド

- 『Oracle8i for Sun SPARC Solaris インストール・ガイド』
- 『Oracle8i for HP 9000 Servers and Workstations インストール・ガイド』
- 『Oracle8i for IBM AIX インストール・ガイド』
- 『Oracle8i for Windows NT インストール・ガイド』
- 『Oracle Diagnostics Pack インストール・ガイド』

オペレーティング・システム固有の管理ガイド

- 『Oracle8i for Sun SPARC Solaris 管理者リファレンス』
- 『Oracle8i for HP 9000 Servers and Workstations 管理者リファレンス』
- 『Oracle8i for IBM AIX 管理者リファレンス』
- 『Oracle Parallel Server for Windows NT 管理者ガイド』
- 『Oracle8i for Windows NT 管理者ガイド』

Oracle Parallel Server Management

- 『Oracle Enterprise Manager 管理者ガイド』
- 『Oracle Diagnostics Pack スタート・ガイド』

Oracle Server マニュアル

- 『Oracle8i 概要』
- 『Oracle8i 管理者ガイド』
- 『Oracle8i リファレンス・マニュアル』
- 『Oracle8i Net8 管理者ガイド』

表記規則

この項では、このマニュアルで使用する次のものに関する表記規則について説明します。

- 本文
- コード例

本文

この項では、本文中で使用される表記規則について説明します。

大文字

大文字のテキストは、コマンド・キーワード、オブジェクト名、パラメータ、ファイル名などに対して注意を促すために使用されます。

たとえば、「プライベート・ロールバック・セグメントを作成する場合、その名前はパラメータ・ファイルの ROLLBACK_SEGMENTS パラメータに含まれる必要があります。」のように使用されます。

コード例

SQL および SQL*Plus のコマンドおよび文は、本文と区別して、固定幅フォントで示されます。たとえば次のとおりです。

```
INSERT INTO emp (empno, ename) VALUES (1000, 'SMITH');  
ALTER TABLESPACE users ADD DATAFILE 'users2.ora' SIZE 50K;
```

例文には、カンマや引用符などの句読点が含まれる場合があります。例文中のすべての句読点は必要です。例文は、すべてセミコロン (;) で終了します。アプリケーションによって、文を終了するためのセミコロンまたはその他の終了記号が、必要な場合と必要でない場合があります。

例文中の大文字語は、Oracle SQL 内のキーワードを示します。ただし、文を発行する場合、キーワードの大 / 小文字は区別されません。

例文中の小文字語は、単なる例として使用されていることを示します。たとえば、小文字語は、表、列またはファイルの名前を示します。

第I部

パラレル処理の基本

第I部では、Oracle Parallel Server プロセスの基本について説明します。第I部に含まれる章は、次のとおりです。

- 第1章「Oracle Parallel Server の概要」
- 第2章「パラレル・ハードウェア・アーキテクチャ」
- 第3章「Oracle Parallel Server のアーキテクチャ」

Oracle Parallel Server の概要

この章では、Oracle Parallel Server のパラレル処理およびパラレル・データベース・テクノロジー機能について説明します。Oracle Parallel Server を使用することによって、オンライン・トランザクション処理 (OLTP)、電子商取引、意思決定支援システム (DSS) およびハイブリッド・システム・タイプにおいて多大な効果が得られます。Oracle Parallel Server の機能を使用して、これらのシステムでパラレル環境の冗長性を効果的に利用できます。

Oracle Parallel Server を使用して、高パフォーマンス、スループットおよび高可用性を得ることもできます。どのような目標においても、これらのテクノロジーを正しく配置して、マルチプロセッシングのパワーの効果を完全に得ることが課題となります。そのためには、Oracle Parallel Server の動作、必要なリソースおよび効果的な使用方法を理解する必要があります。

この章の内容は次のとおりです。

- [Oracle Parallel Server の概要](#)
- [Oracle Parallel Server の効果](#)

注意： Oracle Parallel Server は、パラレル実行機能とは異なります。Oracle Parallel Server は共有データベースにアクセスする複数のコンピュータに関連しますが、パラレル実行はパラレルに操作を実行する複数の処理に関連します。パラレル実行は、単一インスタンスおよび Oracle Parallel Server のインストールの両方に使用できます。

参照： Oracle Parallel Server の詳細は、『Oracle8i 概要』を参照してください。

Oracle Parallel Server の概要

Oracle Parallel Server は、相互接続された複数のコンピュータの処理能力を統合して利用する、強力なコンピューティング環境です。Oracle Parallel Server ソフトウェアおよび「クラスタ」というハードウェアの集まりは、各コンポーネントの処理能力を統合して、単一で強力なコンピューティング環境になります。クラスタは、一般的に 2 つ以上のコンピュータ（または「ノード」）で構成されます。

Oracle Parallel Server 環境では、すべてのノードが同じデータベースに対して同時にトランザクションを実行します。Oracle Parallel Server は、共有データへの各ノードのアクセスを調整し、一貫性および整合性を提供します。

複数のノードのパワーを統合して利用することで、大きな利益が得られます。1 つの大きな作業を下位の作業に分割し、複数のノード間で下位の作業を分散すると、1 つのノードのみで作業する場合よりも早く作業を完了できます。このような種類の平行処理は、順次処理よりも明らかに効率的です。また、パフォーマンスが向上することによって、より大きな作業負荷を処理し、増加するユーザー数に対応することもできます。

正確なパーティション化によってノード対データの高い親和性を確立した場合、アプリケーションを効果的に拡張し、増加するデータ処理の要求に応えることができます。リソースを追加すると、Oracle Parallel Server はそれを利用し、処理能力を個々のコンポーネントの制限以上に拡張できます。

Oracle Parallel Server は、多くのシステム・タイプに使用できます。たとえば、読み専用データにアクセスするデータ・ウェアハウス・アプリケーションは、Oracle Parallel Server の第 1 候補です。さらに、Oracle Parallel Server は、読み専用および読み / 書きのアプリケーションの両方の特性を結合するハイブリッド・システム、および増加するオンライン・トランザクション処理システムを正常に管理します。

Oracle Parallel Server は、強力で高可用性のソリューションにおける重要なコンポーネントとして動作します。適切に構成された Oracle Parallel Server 環境では、最小限の停止時間または停止時間なしで障害に対処できます。

Oracle Parallel Server の効果

次の項では、平行処理の明らかな効果以外の重要な効果について説明します。これらの効果には、単一インスタンス・システム以上のスループットおよびスケーラビリティの改善および応答時間の改善が含まれます。また、Oracle Parallel Server は、クラスタ環境でのノード障害を解決することによって、高可用性の理想的なソリューションも提供します。

Oracle Parallel Server 環境は単一インスタンス環境と機能上同一であるため、単一インスタンス環境と比べた場合、機能上透過的に使用できます。

スケーラビリティ

スケーラビリティとは、適切に配置された Oracle Parallel Server アプリケーションにさらにノードを追加し、パフォーマンスを大幅に向上させることです。Oracle Parallel Server では、機器を追加することによってその効果を得て、複数のシステムの処理能力を利用できます。

高可用性

高可用性とは、ハードウェアまたはソフトウェア障害が発生しても、一貫性のある連続的なサービスを提供する、冗長なコンポーネントを持つシステムを示します。最も高可用性な構成では、ノードは他のノードから孤立しているため、1つのノードの障害がシステム全体に影響を及ぼすことはありません。この場合、障害のないノードが障害ノードをリカバリし、システムは継続してユーザーにデータ・アクセスを提供します。これは、単一ノードでノード障害が発生した場合よりも、データは一貫して使用可能になることを意味します。高可用性は、データベースの可用性の増加も意味します。

透過性

透過性とは、単一インスタンスの排他モードで動作する Oracle と、共有モードで動作する Oracle Parallel Server が機能的に等価であることです。単一インスタンスの Oracle 上で動作するアプリケーションは、Oracle Parallel Server を使用しても同様に動作します。Oracle データベースは、次の3つの異なるモードで実行するように構成できます。

- 単一インスタンスでの排他モード
- 単一インスタンスでの共有モード
- 2つ以上のインスタンスでの共有モード

共有モードの複数ノードからトランザクションを実行する場合、Oracle Parallel Server オプションをインストールする必要があります。Oracle Parallel Server は、単一インスタンス環境で使用できる機能以外に、多くのパフォーマンス機能を提供します。

Oracle Parallel Server の高パフォーマンス機能

Oracle Parallel Server では、Oracle 固有のトランザクション処理機能を犠牲にすることなく、クラスタ構成でのパラレル処理の効果を得られます。次の項では、排他モードおよび共有モードでの Oracle の特定の機能について説明します。これらの機能によって、Oracle Parallel Server を使用してアプリケーションを実行すると、アプリケーションのパフォーマンスが向上します。

バッファ・キャッシュ管理 Oracle は、単一インスタンス内では、データ・ブロックやロック情報などのリソースを、メモリー内に常駐するバッファ・キャッシュに格納します。この情報をローカルに格納することで、データベース処理に必要なディスク I/O の量を削減します。Parallel Server の各ノードは他のノードと共有しない独自のメモリーを持つため、Oracle Parallel Server は、パフォーマンスを低下させる余分なディスク I/O を最小化すると同時に、異なるノード間のバッファ・キャッシュを調整する必要があります。Oracle のパラレル・キャッシュ管理テクノロジーは、複数のバッファ・キャッシュを調整すると同時に、Oracle の高パフォーマンス機能を保持します。

参照： バッファ・キャッシュの詳細は、『Oracle8i 概要』を参照してください。

高速コミット、グループ・コミットおよび遅延書き込み 高速コミット、グループ・コミットおよび遅延書き込みは、Oracle の各インスタンス・ベースで操作し、排他モードでも共有モードでも同様に動作します。

データ・ブロックが、データを要求するインスタンスのバッファ・キャッシュにない場合のみ、そのデータ・ブロックがディスクから読み込まれます。データ・ブロック書き込みが遅延されるため、データ・ブロックには、よく複数のトランザクションからの変更が含まれません。

最適な状態では、次のような必要な場合のみ、変更データ・ブロックがディスクに書き込まれます。

- ブロックがしばらく使用されていなかった場合、および新しいデータ用にバッファ・キャッシュ領域が必要な場合（共有モードまたは排他モード）
- チェックポイント中（共有モードまたは排他モード）
- 他のインスタンスでブロックが必要な場合（共有モードのみ）

行ロックおよび複数バージョンの読取り一貫性 Oracle の行ロック機能によって、別のノードからの複数のトランザクションで、同じデータ・ブロックの異なる行をロックおよび更新できます。これは、他のトランザクションがコミットされるのを待つことなく、実行されます。行が変更され、まだコミットされていない場合、元の行の値を、すべてのインスタンスで読み込むことができます。これを、「複数バージョンの読取り一貫性」といいます。

オンライン・バックアップおよびアーカイブ Oracle Parallel Server は、排他モードで使用できるすべての Oracle バックアップ機能をサポートしています。これには、データベース全体または個々の表領域の、オンライン・バックアップおよびオフライン・バックアップの両方が含まれます。

ARCHIVELOG モードで Oracle を操作する場合、オンライン REDO ログ・ファイルは、上書きされる前にアーカイブされます。Oracle Parallel Server では、各インスタンスがその REDO ログ・ファイルを自動的にアーカイブできます。または、1 つ以上のインスタンスが、すべてのインスタンスの REDO ログ・ファイルを手動でアーカイブできます。

ARCHIVELOG モードでは、オンライン・バックアップおよびオフライン・バックアップの両方が作成できます。Oracle を NOARCHIVELOG モードで操作する場合、オフライン・バックアップしか作成できません。データの損失を防ぐために、本番データベースを ARCHIVELOG モードで操作することをお勧めします。

パラレル・ハードウェア・アーキテクチャ

この章では、クラスタ環境の特徴を表すハードウェア・コンポーネントおよび様々な高レベルのアーキテクチャ・モデルについて説明します。Oracle Parallel Server アプリケーションを配置するために選択するモデルは、処理の目的によって異なります。

Oracle Parallel Server 環境は、通常、クラスタを形成するために相互接続されたいくつかのノードで配置されます。この章では、ノード用の基本ハードウェアおよびクラスタにノードを作成するために使用されるハードウェアについて説明します。

この章の内容は次のとおりです。

- クラスタ・ハードウェア・コンポーネントの概要
- メモリー・アクセス
- 高速インターコネクト
- クラスタ / ノードおよびインターコネクト
- クラスタ化システムでのストレージ・アクセス
- 様々なクラスタ上で動作する Oracle Parallel Server

クラスタ・ハードウェア・コンポーネントの概要

クラスタは、インターコネクでリンクされた2つ以上のノードから構成されます。インターコネクは、クラスタ内のノード間の通信経路として機能します。ノードは、各インスタンスの共有データの操作を同期化するのに必要な通信に対してインターコネクを使用します。ノードがアクセスする共有データは、記憶デバイス内に常駐します。クラスタを「疎結合コンピュータ・システム」ともいいます。

次の項では、これらのコンポーネントをさらに詳しく説明します。

ノードの概要

ノードには、主に次の4つのコンポーネントがあります。

- CPU-コンピュータのメイン・メモリーからの読み込み、またはメイン・メモリーへの書き込みを行う、コンピュータのメイン処理コンポーネントです。
- メモリー-プログラムの実行およびデータのバッファに使用されるコンポーネントです。
- ストレージ-データを格納するデバイスです。通常は、その内容を変更するために読み込み/書き込みトランザクションでアクセスする必要がある永続ストレージです。
- インターコネク-ノード間の通信リンクです。

ユーザーは様々な構成で、これらのコンポーネントを購入できます。その構成によって、クラスタ内の各ノードがどのようにメモリーおよびストレージにアクセスするかが決定します。

すべてのクラスタは、多かれ少なかれ同じ方法でCPUを使用します。ただし、残りのコンポーネント、メモリー、ストレージおよびインターコネクは、様々な目的に対して様々な方法で構成できます。この章の以降の項では、次の項目を説明し、クラスタがこれらのコンポーネントをどのように使用するかについて説明します。

- [メモリー・アクセス](#)
- [高速インターコネク](#)
- [クラスタ / ノードおよびインターコネク](#)
- [クラスタ化システムでのストレージ・アクセス](#)

メモリー・アクセス

通常複数のCPUがメイン・メモリーを共有するよう構成されます。これによって、スケラブルなパフォーマンスを実現する単一コンピュータ・システムを作成できます。また、このタイプのシステムは、同等の処理能力がある単一のCPUよりも低いコストで作成できます。単一のCPUを持つコンピュータを「ユニプロセッサ」といいます。

共有メモリー・システムの構成には、2種類あります。

- 均一メモリー・アクセス
- 不均一メモリー・アクセス

共有メモリー・システムを「密結合コンピュータ・システム」といいます。

均一メモリー・アクセス

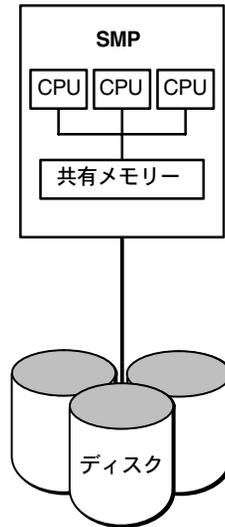
均一メモリー・アクセス構成（UMA）では、すべてのプロセッサは同じスピードでメイン・メモリーにアクセスできます。この構成では、メモリー・アクセスは均一です。この構成を「対称マルチプロセッシング・システム（SMP）」ともいいます。

不均一メモリー・アクセス

不均一メモリー・アクセス（NUMA）は、すべてのプロセッサがすべてのメモリー構造にアクセスできることを意味します。ただし、メモリー・アクセスは等価ではありません。つまり、アクセス・コストは、各プロセッサがメモリーのどの部分にアクセスするかによって変わります。NUMA 構成では、メイン・メモリーの特定の場所にアクセスするコストは、その他の場所に関連するいくつかの CPU で異なります。

UMA/SMP および NUMA システムにおけるパフォーマンスは、両方ともメモリー・バス帯域幅によって制限されます。これは、ある点を越えて CPU をシステムに追加しても、パフォーマンスは比例して増加しないことを意味します。CPU の追加によってパフォーマンスの向上が最小限になる点は、アプリケーション・タイプおよびシステム・アーキテクチャによって変わります。通常、SMP 構成は、24 ~ 64 プロセッサを越えると拡張性が非常に低下します。

図 2-1 密結合共有メモリー・システムまたは SMP/UMA



共有メモリーのメリット

共有メモリー・システムの並列処理のメリットは、次のとおりです。

- メモリー・アクセスは、疎結合システムでのアクセスよりもコストがかかりません。
- 共有メモリー・システムは、クラスタよりも管理が簡単です。

並列処理に対する共有メモリー・システムのデメリットは、バスの帯域幅および待ち時間、また使用可能なメモリーによってスケーラビリティが制限されることです。

高速インターコネクト

高速インターコネクトは、クラスタ内で各ノードから他のノードに接続する、高帯域幅で待ち時間の少ない通信機能です。高速インターコネクトは、ノード間でメッセージおよび他の並列処理固有の通信を送信し、データおよびデータ依存リソースへの各ノードのアクセスを調整します。

Oracle Parallel Server は、ユーザー・モードのプロセス間通信 (IPC) および「メモリー・マップド IPC」も使用します。これによって、CPU の消費を大幅に削減し、IPC の待ち時間も削減します。

インターコネクには、イーサネット、FDDI (Fiber Distributed Data Interface) または他の独自のハードウェアを使用できます。また、プライマリ・インターコネクに障害が発生した場合に使用できる、インターコネクのバックアップが必要です。インターコネクのバックアップを準備すると、可用性が向上し、インターコネクが原因の1つになって起こりうる障害を削減できます。

クラスタ / ノードおよびインターコネク

前述のとおり、ユニプロセッサ、SMP または NUMA メモリー構成のいずれかを使用する必要があります。インターコネクで構成する場合、これらのタイプの2つ以上のプロセッサでクラスタを構成します。クラスタ化システムのパフォーマンスは、多くの要因によって制限されます。これには、メモリー帯域幅、CPU 間通信の帯域幅、システムに使用可能なメモリー、I/O 帯域幅、インターコネク帯域幅などの様々なシステム・コンポーネントが含まれます。

クラスタ化システムでのストレージ・アクセス

クラスタ化システムでは、いくつかのアーキテクチャ・モデルが使用されます。各アーキテクチャは、特定の目的に対して最適な方式を共有する、特定のリソースを使用します。

この項では、次のアーキテクチャについて説明します。

- [均一ディスク・アクセス](#)
- [不均一ディスク・アクセス](#)

ストレージ・アクセスのタイプは、メモリー・アクセスのタイプに依存しません。たとえば、SMP ノードのクラスタは、均一または不均一のいずれのディスク・サブシステムでも構成できます。

均一ディスク・アクセス

[図 2-2](#) に示す均一ディスク・アクセス・システムまたは共有ディスク・システムでは、ディスク・アクセスのコストはすべてのノードに対して同じです。

図 2-2 均一アクセス共有ディスク・システム

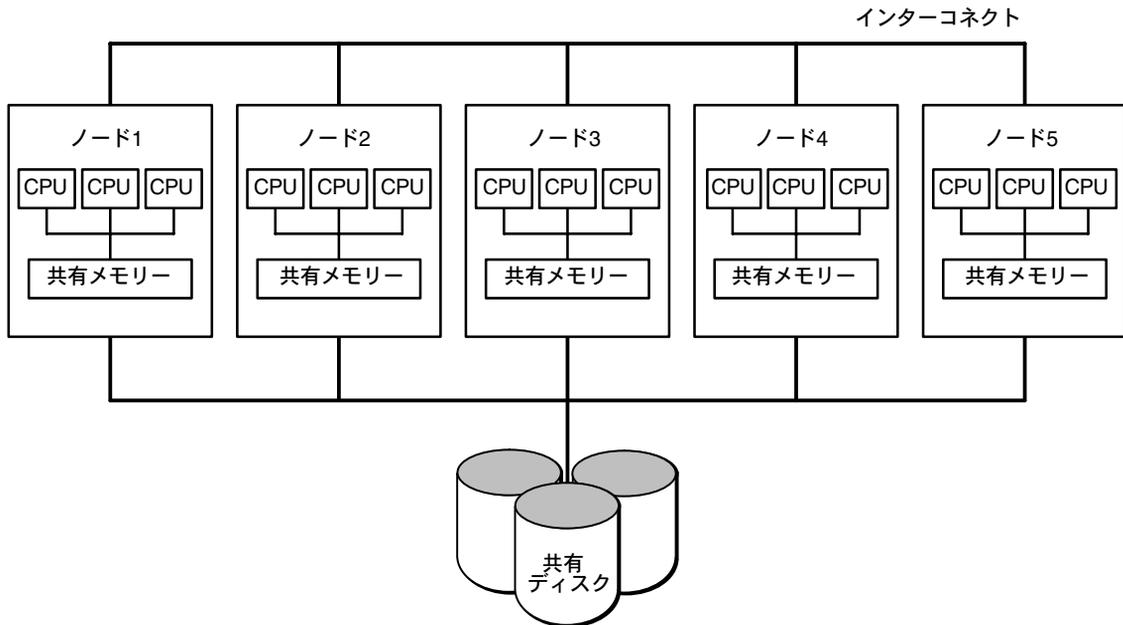


図 2-2 のクラスタは、複数の SMP ノードで構成されています。このような共有ディスク・サブシステムは、多くの場合、ディスク・ファームへの SCSI または Fibre Channel 接続を使用して実装されます。

共有ディスク・システムで平行処理を使用するメリットは、次のとおりです。

- 共有ディスク・システムでは、高可用性が得られます。1つのノードに障害が発生した場合でも、すべてのデータにアクセス可能です。
- 共有ディスク・システムは、段階的な拡張を可能にします。

不均一ディスク・アクセス

いくつかのシステムでは、ディスク・ストレージは1つのノードにのみ連結されます。このノードでは、アクセスはローカルです。他のすべてのノードでは、ディスク・アクセスおよびデータの要求は、ソフトウェアの仮想ディスク・レイヤーによって、ディスクがローカルに連結されたノードへインターコネクトを介して転送される必要があります。これは、ディスクの読み込みまたは書き込みのコストが、アクセスがローカルかリモートかによって大きく変わることを意味します。インターコネクトの待ち時間および IPC のオーバーヘッドを含む、リモート・ディスクからのブロックの読み込みまたは書き込みに関するコストの分だけ、このタ

イブの操作のコストが、均一ディスク・アクセス構成を使用した同じタイプの操作のコストと比べて大きくなります。

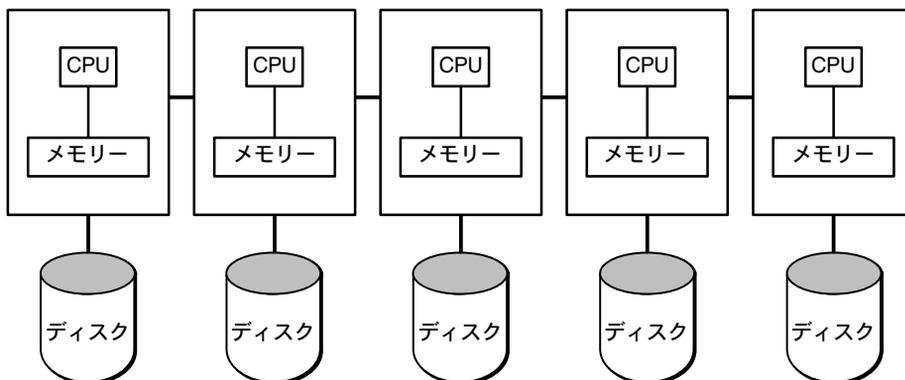
不均一ディスク・アクセス構成は、一般に、「シェアード・ナッシング・システム」または「超並列（MPP）システム」というシステムで使用されます。高可用性を得るために、1つのノードに障害が発生した場合、通常、そのローカル・ディスクは別のノードに対するローカル・ノードになるように再構成できます。これらの不均一ディスク・アクセス・システムに対して、Oracle Parallel Server では、仮想ディスク・レイヤーがシステム・レベルで提供される必要があります。場合によっては、ディスクまたは他の I/O デバイスがローカルに連結されたノードへ作業を移動させる方が、リモート要求を使用するよりも非常に効率的です。この処理とストレージを連結することを「ディスク親和性」といい、Oracle では、パラレル実行およびバックアップを含む様々な領域で使用されています。

MPP または不均一ディスク・アクセス・システム上でパラレル処理を使用するメリットは、次のとおりです。

- ノードの数は、ディスクへの物理的な接続数によって制限されません。
- ノードを追加できるため、ディスク・ストレージの総量を非常に大きくできます。

図 2-3 に、シェアード・ナッシング・システムを示します。

図 2-3 不均一ディスク・アクセス



様々なクラスタ上で動作する Oracle Parallel Server

Oracle Parallel Server は、多くのベンダーの様々なクラスタ化システムでサポートされています。アーキテクチャの面では、Oracle Parallel Server がサポートできるクラスタ内のノードの数は、既知のいかなる実装よりも大幅に多くなっています。主に高可用性を得るために構成された小さなシステムでは、クラスタ内に 2 つのノードしか存在しない場合があります。ただし、大規模な構成では、クラスタ内に 40 ~ 50 のノードが存在する場合があります。一般に、クラスタ管理のコストは、システム内のノードの数に関連しています。最近では、共有ディスクを使用して大規模な SMP システムでノードを構成し、そのノードをより少ない数で使用する傾向があります。

Oracle Parallel Server のアーキテクチャ

この章では、Oracle Parallel Server の処理に対して提供される、アーキテクチャ・コンポーネントについて説明します。これらのコンポーネントは、単一インスタンスに対するコンポーネントに追加されるものです。したがって、Oracle Parallel Server 特有のものです。いくつかのコンポーネントは Oracle ソフトウェアで提供され、その他はベンダー固有のものです。

この章の内容は次のとおりです。

- クラスタ化システムに対する Oracle Parallel Server コンポーネント
- Cluster Manager
- 分散ロック・マネージャ
- クラスタ・インターコネクトおよびプロセス間通信（ノード間）
- ディスク・サブシステム

参照： Oracle の単一インスタンスのアーキテクチャ・コンポーネントの詳細は、『Oracle8i 概要』を参照してください。

クラスタ化システムに対する Oracle Parallel Server コンポーネント

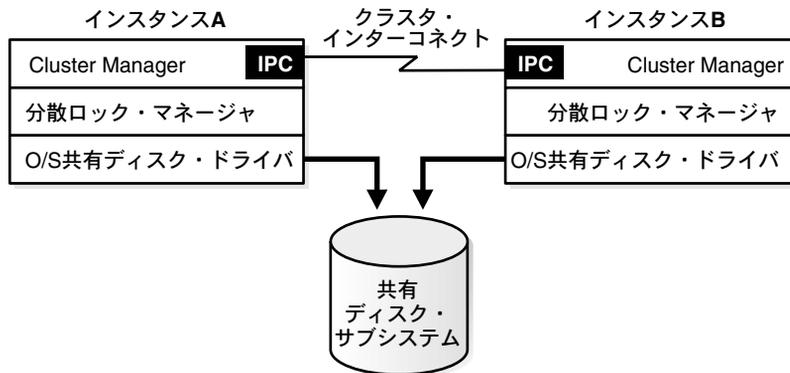
第2章で説明するアーキテクチャ・モデルに基づいて、Oracle Parallel Server の実装に必要なソフトウェアについて次に説明します。

各ハードウェア・ベンダーは、オペレーティング・システム固有のレイヤーを使用して、パラレル処理を実装します。これらのレイヤーは、オペレーティング・システムと、この章で説明する Oracle Parallel Server ソフトウェアとの間の通信リンクとして動作します。

クラスタ化システムに対するコンポーネントの概要

図 3-1 に、これらのコンポーネントのおおまかな図を示します。

図 3-1 パラレル処理に対するクラスタ・コンポーネント



Cluster Manager ソフトウェアは、インターコネクトを介して転送されるノード間のメッセージ機能を監視し、ノード間の操作を調整します。分散ロック・マネージャは、パラレル・キャッシュ管理機能の操作を監視します。次に、次のコンポーネントについて詳しく説明します。

- Cluster Manager
- 分散ロック・マネージャ
- クラスタ・インターコネクトおよびプロセス間通信（ノード間）
- ディスク・サブシステム

Cluster Manager

Cluster Manager は、クラスタ、およびそのクラスタ内にあるすべてのノードのグローバル・ビューを提供します。また、クラスタ・メンバーシップも制御します。通常、Cluster Manager は、ベンダーによって提供されるコンポーネントです。ただし、Oracle では、Windows NT 環境用の Cluster Manager が提供されています。

Oracle Parallel Server も、Cluster Manager と協調して高可用性を実現します。Cluster Manager は、インスタンスが起動および停止したときに、自動的に起動および停止します。

障害検出

Cluster Manager の切断は、Cluster Manager のクライアントが自発的に切断した場合、クライアントのプロセスが終了した場合、またはクライアントのノードが停止するか障害が起こった場合に発生します。1 つ以上のノードに障害が発生した場合も同じです。Cluster Manager が、ノードが非アクティブ、または適切に機能していないと判断すると、Cluster Manager はノードまたはインスタンス上のすべてのプロセスを終了します。

障害があった場合、リカバリはユーザー・アプリケーションに対して透過的です。Cluster Manager は、自動的にシステムを再構成して障害ノードを孤立させた後、分散ロック・マネージャに状態を通知します。その後、Oracle Parallel Server がデータベースを有効状態にリカバリします。

Node Monitor

Cluster Manager には、「Node Monitor」という機能があります。Node Monitor は、ノード、インターコネクト・ハードウェアとソフトウェア、共有ディスクおよび Oracle インスタンスを含む、クラスタ内の様々なリソースの状態をポーリングします。Cluster Manager およびその Node Monitor がこれらの操作を実行する方法は、オペレーティング・システム固有のレイヤーの Oracle の実装に基づきます。

Cluster Manager は、クラスタ内のリソースの状態が変更されたとき、クライアントおよび Oracle Server に知らせます。たとえば、別のデータベース・インスタンスが Cluster Manager に登録された場合、またはインスタンスが Cluster Manager から切断された場合に、Oracle Server はこれを認識する必要があります。

前述のとおり、Cluster Manager は、ノード、ネットワークおよびインスタンスを含む、様々なクラスタ・リソースの状態を監視しています。また、Node Monitor は、Cluster Manager に対して次の作業も行います。

- クラスタ環境で Oracle Parallel Server に必要な、基本ノード管理インタフェース・モジュールを提供します。
- クラスタ全体にクラスタ・メンバーシップの共通ビューを提供することによって、ノードのメンバーシップ状態を検出および追跡します。
- カレント・メンバーシップのすべてのノードを問い合わせることによって、すべてのノード上で動作し、かつクラスタのトポロジを監視します。
- 変更のイベントを通知しているアクティブ・ノードでの状態変更を検出および診断し、すべてのノード間で、新しく共通で一貫性のある状態を調整します。
- Oracle Parallel Server にクラスタ・メンバーシップの変更を通知します。

参照： 高可用性の詳細は、[第 9 章](#)を参照してください。

分散ロック・マネージャ

分散ロック・マネージャは、共有データベースおよびデータベース内の共有リソースへの同時アクセスを調整する、Parallel Server の統合されたコンポーネントです。これによって、一貫性およびデータ整合性が保持されます。この項では、次の分散ロック・マネージャの機能について説明します。

- [透過性](#)
- [分散アーキテクチャ](#)
- [フォールト・トレラント](#)
- [リソースのマスター化](#)
- [デッドロック検出](#)
- [永続リソース](#)

透過性

分散ロック・マネージャによって実行されるリソースへのアクセス調整は、アプリケーションに対して透過的です。アプリケーションは、単一インスタンス環境で使用されるものと同じロック・メカニズムを継続して使用します。

分散アーキテクチャ

分散ロック・マネージャは、ロック・データベースをメンテナンスし、リソースおよびそのリソースが保持するロックの情報を記録します。ロック・データベースはメモリーに常駐し、クラスタ全体のすべてのノードに対して分散されます。この分散アーキテクチャでは、各ノードがグローバル・ロック管理の一端を担い、グローバル・ロック・データベースの一部を管理します。この分散ロック管理方式は、フォールト・トレラントを提供し、実行時のパフォーマンスを向上させます。

フォールト・トレラント

分散ロック・マネージャは、複数のノードに障害が発生した場合でも、継続的なサービスを提供し、ロック・データベースの整合性を保持するフォールト・トレラントなサービスです。共有データベースは、リカバリ完了後、少なくとも1つのインスタンスがデータベースでアクティブである限りアクセスできます。

また、フォールト・トレラントは、Oracle Parallel Server 内のインスタンスが、いつでも、どのような順序でも、起動および停止できるようにします。ただし、インスタンスを再構成すると、わずかな処理の遅延の原因となる場合があります。

リソースのマスター化

分散ロック・マネージャは、特定のリソースにアクセスする際に必要となるロックに関する情報をすべてのノードに分散してメンテナンスします。通常、分散ロック・マネージャは、1つのノードを指定し、リソースおよびそのロックに関するすべての情報を管理します。

Oracle Parallel Server は、静的ハッシング・ロック・マスター化方式を使用します。このマスター化処理によって、リソースに対するマスターとして動作する Parallel Server インスタンスの1つに対して、リソースの名前をハッシュします。これによって、すべての使用可能なノード間でのリソースの配布が、均等で任意になります。各リソースは、特定のマスター・ノードに対応付けられます。

分散ロック・マネージャは、各状況で使用されるロック・マスター化の方法を最適化します。このロック・マスター化の方法は、インスタンス起動時および通常の実行アクティビティ中のシステム・パフォーマンスに影響します。リソースがローカルにマスター化された場合、パフォーマンスは最適化されます。

デッドロック検出

分散ロック・マネージャは、デッドロックの影響を受けやすいすべてのロックおよびリソースに対して、デッドロック検出を実行します。データベース自体の表またはオブジェクトへのアクセスは制御しません。Oracle Parallel Server は、分散ロック・マネージャを使用して、リソース（データ・ブロックやロールバック・セグメントなど）への複数のインスタンスにわたる同時アクセスを調整します。

永続リソース

分散ロック・マネージャは、永続リソースを提供します。リソースは、ロックを保持しているすべてのプロセスまたはグループが異常終了した場合でも、その状態を保持します。

DLM 処理の例

クラスタ内のあるノードで、データベース内のブロック番号 n を変更する必要があるとします。同時に、別のノードで同じブロックを更新し、トランザクションを完了させる必要がありますとします。

分散ロック・マネージャがない場合、両方のノードは同じブロックを同時に更新します。分散ロック・マネージャがある場合は、1つのノードのみがブロックを更新できます。もう一方のノードは待機する必要があります。分散ロック・マネージャによって、ある時点では、1つのインスタンスのみがブロック更新の権利を持つことが保証されます。これによって、すべての変更が一貫性のある方法で保存されることが保証され、データの整合性が得られます。

Cluster Manager との相互作用

分散ロック・マネージャは、Cluster Manager から独立して動作します。分散ロック・マネージャは、Cluster Manager に依存して、他のノードの状態に関する適時で正確な情報を得ています。分散ロック・マネージャは、クラスタ内の特定のインスタンスから必要な情報が得ることができない場合、そのインスタンスを停止します。これによって、各インスタンスが他のすべてのインスタンスを考慮してディスク・アクセスを調整する必要があるため、Oracle Parallel Server データベースの整合性が保証されます。

クラスタ・インターコネクトおよびプロセス間通信（ノード間）

Oracle Parallel Server の機能上の最大のメリットは、相互接続された複数のマシン上で実行できることにあります。これを容易にするために、Oracle Parallel Server は、基礎となるプロセス間通信（IPC）コンポーネントに大きく依存しています。

IPC は、Oracle Parallel Server 環境のインスタンス間でメッセージを転送するために必要なプロトコルおよびインタフェースを定義します。メッセージは、このインタフェースにおける通信の基本単位です。核となる IPC 機能は、非同期のキュー・メッセージング・モデルを中心にして構築されています。IPC は、個別のメッセージを、ハードウェアが許す限り高速に、送信および受信するよう設計されています。各種サービスは、最適化された通信レイヤーの上に実装できます。このようにして、分散ロック・マネージャはその通信を実行します。

ディスク・サブシステム

オペレーティング・システム固有のレイヤーに加えて、Oracle Parallel Server では、すべてのノードがディスクに同時アクセスできる必要もあります。これによって、複数のインスタンスが同じデータベースに同時にアクセスできます。

第II部

Oracle Parallel Server のロック処理

第II部では、Oracle Parallel Server が実行するロック処理について説明します。これによって、データ・アクセスが同期化され、データの整合性が保証されます。第II部に含まれる章は、次のとおりです。

- [第4章「インスタンス間の調整」](#)
- [第5章「パラレル・キャッシュ管理」](#)

4

インスタンス間の調整

この章では、クラスタ内で発生するインスタンス間の調整アクティビティについて詳しく説明します。第3章で説明するとおり、Oracle は、クラスタ内でロックを使用して、ロック・リソース、データおよびインスタンス間のデータ要求を調整します。この章では、Oracle がこれらのリソースを調整する方法について詳しく説明します。

この章の内容は次のとおりです。

- 同期化
- ローカル・ロック
- グローバル・ロック
- 非パラレル・キャッシュ管理の調整
- パラレル・キャッシュ管理の調整

同期化

クラスタ内の同時作業を調整することを「同期化」といいます。データ・ブロックやロックなどのリソースは、クラスタ内のノードがそれらのオーナーシップを取得および解放するときに、同期化される必要があります。Oracle Parallel Server が提供する同期化によって、リソースのクラスタ全体の同時実行性が保持され、それによって、共有データの統合性が保証されます。

パラレル処理を正常に行うには、同期化がほとんど必要ないように、ノード間のリソースを必要とする作業を分割することが重要です。必要な同期化が少ないほど、システムはスピードアップおよびスケールアップします。過剰なノード間通信が必要な場合は、同期化のオーバーヘッドが非常に大きくなります。

ノード間でパラレル処理を実行するために行う同期化では、高速インターコネクトを使用してパラレル処理を行うプロセスをリンクすることが理想的です。1つのノード内でのパラレル処理の場合、メッセージ機能は必要ではありません。かわりに、共有メモリーが使用されます。前の章で説明するとおり、ノード間のメッセージ機能およびロック機能は、分散ロック・マネージャによって処理されます。

同期化の量は、リソースの量、およびそのリソースで動作するユーザーおよび作業の数に依存します。少数の同時作業の調整には同期化はあまり必要ありませんが、同時作業の数が多い場合は、かなりの同期化が必要です。

Oracle Parallel Server の各インスタンスには、そのシステム・グローバル領域にデータ・ディクショナリ情報を含んだ、ディクショナリ・キャッシュまたは行キャッシュがあります。Oracle Parallel Server の Oracle インスタンスのデータ・ディクショナリ構造は、排他モードでのインスタンスのものと同じです。Parallel Server は、グローバル・ロックを使用して、複数のインスタンス間のデータ・ディクショナリ・アクティビティを調整します。

パラレル・キャッシュ管理では、いくつかのタイプのロックを使用して、クラスタ内のデータおよびリソースへのアクセスが制御されます。使用するロックおよび細分性の程度を構成する方法は、Oracle Parallel Server で実行するアプリケーションのパフォーマンスに影響を及ぼします。細分性の程度とは、インスタンスごとのロック数を示します。

アプリケーションを適切に設計することによって、各タイプのロックに最適な影響を与えることができます。次に、初期化パラメータを設定し、クラスタを適切に管理することによっても、システムのオーバーヘッドに適切な影響を与えます。一方、不適切にロックを使用すると、システムが共有リソースを同期化するために非常に時間がかかり、スピードアップまたはスケールアップすることができません。

Oracle Parallel Server では、次の項で説明するように、2つの主要なグループのロックが使用されます。

- ローカル・ロック
- グローバル・ロック

ローカル・ロック

ローカル・ロックには、ラッチおよびエンキューの2つのタイプがあります。ラッチは、Parallel Server 固有ではなく、各インスタンス内でのみ同期化されます。したがって、ラッチは、クラスタ環境のグローバル操作に影響を及ぼしません。ただし、エンキューは、インスタンスに対してローカルにも、クラスタに対してグローバルにもなれます。

次に、この2つのタイプのロックについて簡単に説明します。

- ラッチ
- エンキュー

ラッチ

ラッチは、システム・グローバル領域のメモリー内部のデータ構造を保護する、単純で低レベルのシリアライズ化メカニズムです。ラッチは、データ・ファイルなどは保護せず、自動で、排他モードで非常に短い時間のみ保持されます。前述のとおり、ラッチは、1つのノード内で同期化されるため、ノード間の同期化には役立ちません。

エンキュー

エンキューは、データベース・リソースへのアクセスをシリアライズ化する共有メモリー構造です。Parallel Server を使用可能にしない場合、エンキューは1つのインスタンスに対してローカルになります。Parallel Server を使用可能にした場合、エンキューはデータベースに対してグローバルになります。エンキューは、セッションまたはトランザクションに対応付けられ、Oracle は、それらのエンキューを次のいずれかのモードで使用できます。

- 共有またはプロテクト読み
- 排他
- プロテクト書き込み
- 同時読み
- 同時書き込み
- NULL

エンキューは、ラッチよりも長く保持されます。また、細分性およびモードはラッチよりも多く、より多くのデータベース・リソースを保護します。たとえば、表ロックまたは DML ロックを要求した場合、その要求はエンキュー・ロックとして割り当てられます。エンキューは、分散ロック・マネージャによって管理されます。Parallel Server が使用可能な場合、ほとんどのローカル・エンキューはグローバル・エンキューになります。

グローバル・ロック

Oracle Parallel Server は、クラスタ内のすべてのアクティブ・インスタンス間でグローバル・ロックを同期化します。グローバル・ロックには、主に次の 2 つのタイプがあります。

- 分散ロック・マネージャによって、パラレル・キャッシュ管理に対して使用されるロック
- 非パラレル・キャッシュ管理リソースを調整するために、クラスタ内で同期化される、グローバル・エンキューなどのグローバル・ロック

分散ロック・マネージャは、すべての Oracle ロック・メカニズムの状態を追跡します。Parallel Server を使用可能にして Oracle インスタンスを起動した場合のみ、グローバル・ロックが作成されます。これに対する例外は、マウント・ロックです。分散ロック・マネージャは、ロック・リソースの状態を Oracle Parallel Server クラスタ内のすべてのインスタンスに対して通信することによって、グローバル・ロックを同期化します。

グローバル・ロックは、トランザクションによってではなく、インスタンス内のバックグラウンド・プロセスによって保持されます。インスタンスは、データ・ブロックやデータ・ディクショナリ・エントリなどのリソースがインスタンスのシステム・グローバル領域に入ったとき、そのリソースを保護するグローバル・ロックを所有します。分散ロック・マネージャは、複数のインスタンスによってアクセスされるリソースに対してのみ、ロックを管理します。

この章の以降の項は次のとおりです。

- [非パラレル・キャッシュ管理の調整](#)
- [パラレル・キャッシュ管理の調整](#)

非パラレル・キャッシュ管理の調整

非パラレル・キャッシュ管理には、データ・ブロック以外のリソースの調整が含まれます。Oracle Parallel Server のパラレル・キャッシュ管理機能については、この章の後半で説明します。

非パラレル・キャッシュ管理ロック

非パラレル・キャッシュ管理ロックには、様々なタイプがあります。これらは、データ・ファイルおよび制御ファイルへのアクセスを制御します。また、ライブラリおよびディクショナリ・キャッシュも制御し、インスタンス間で様々なタイプの通信を実行します。これらのロックは、データ・ファイル・ブロックを保護しません。これらの例としては、DML エンキュー（表ロック）、トランザクション・エンキュー、および DDL ロックまたはディクショナリ・ロックがあります。システム変更番号（SCN）およびマウント・ロックは、エンキューではなく、グローバル・ロックです。

注意： Oracle Parallel Server のコンテキストによって、ほとんどのローカル・エンキューがグローバルになります。それらは、依然、V\$LOCK などのエンキューを表示する固定表およびビューに表示されます。ただし、V\$LOCK 表には、SCN ロック、マウント・ロック、パラレル・キャッシュ管理ロックなどのグローバル・ロックは表示されません。

非 PCM グローバル・ロック

この項では、いくつかの最も一般的な非 PCM グローバル・ロックについて説明します。内容は次のとおりです。

- [非パラレル・キャッシュ管理ロックの概要](#)
- [トランザクション・ロック](#)
- [表ロック](#)
- [ライブラリ・キャッシュ・ロック](#)
- [ディクショナリ・キャッシュ・ロック](#)
- [データベース・マウント・ロック](#)

参照： 分散ロック・マネージャで構成する、非 PCM リソースおよびロックの数の計算方法の詳細は、『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を参照してください。

非パラレル・キャッシュ管理ロックの概要

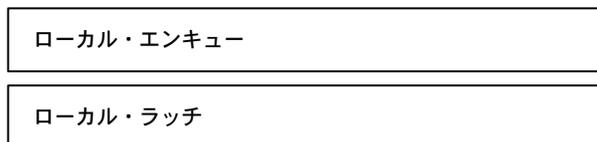
この項では、Oracle が非 PCM ロックを使用して、Oracle 環境でトランザクション、表およびその他のエンティティのロックを管理する方法について説明します。図 4-1 に、Oracle で使用される他のロックと関連する、非 PCM ロックを示します。

図 4-1 Oracle ロック・メカニズム：非 PCM ロック

インスタンス・ロック



ローカル・ロック



PCM ロックは静的ですが（アプリケーションを設計する際、それらを割り当てます）、非 PCM ロックは非常に動的です。システムの初期化パラメータ値が変更されると、非 PCM ロックの数および対応する領域要件も変更されます。

参照： すべての非 PCM ロックの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

トランザクション・ロック

行ロックは、選択された行を保護するロックです。トランザクションは、次のいずれかの文で変更された、各行のグローバル・エンキューおよび排他ロックを取得します。

- INSERT
- UPDATE
- DELETE
- FOR UPDATE 句を含む SELECT

これらのロックはブロックに格納され、各ロックはグローバル・トランザクション・エンキューを参照します。

トランザクションが最初の変更を開始すると、トランザクション・ロックが排他モードで取得されます。これは、トランザクションがコミットまたはロールバックされるまで保持されます。SMON も、トランザクションをリカバリ（取消し）するときに、トランザクション・ロックを排他モードで取得します。トランザクション・ロックは、進行中のトランザクションによってロックされたオブジェクトの解放を待つ、プロセスのキューイング・メカニズムとして使用されます。

表ロック

表ロックは、表全体を保護する DML ロックです。表が、INSERT 文、UPDATE 文、DELETE 文、FOR UPDATE 句を含む SELECT 文および LOCK TABLE 文のいずれかで変更される場合、トランザクションは表ロックを取得します。表ロックは、NULL (N)、行共有 (RS)、行排他 (RX)、共有ロック (S)、共有行排他 (SRX) および排他 (X) のいずれのモードでも保持されます。

ライブラリ・キャッシュ・ロック

SQL (DML/DDDL) 文、PL/SQL 文または Java 文の解析またはコンパイル中に、データベース・オブジェクト（表、ビュー、プロシージャ、ファンクション、パッケージ、パッケージ本体、トリガー、索引、クラスタ、シノニム）が参照された場合、文の解析またはコンパイルのプロセスで、カレント・モードのライブラリ・キャッシュ・ロックが取得されます。Oracle8 では、解析またはコンパイルが終了するまで（解析コールの存続期間）のみ、ロックが保持されます。

ディクショナリ・キャッシュ・ロック

データ・ディクショナリ・キャッシュには、データ・ディクショナリ（メタデータ・ストア）からの情報が含まれています。このキャッシュによって、データ・ディクショナリへの効率的なアクセスが提供されます。

たとえば、新しい表を作成すると、その表のメタデータは、データ・ディクショナリにキャッシュされます。表が削除される場合、メタデータはデータ・ディクショナリ・キャッシュから削除される必要があります。データ・ディクショナリ・キャッシュへのアクセスを

同期化するために、ラッチが排他モードおよび単一共有モードで使用されます。グローバル・ロックは、複数共有（パラレル）モードで使用されます。

Oracle Parallel Server では、1つのインスタンスで削除された表のメタデータが、すべてのノードのデータ・ディクショナリ・キャッシュに含まれている場合があります。この表のメタデータは、すべてのインスタンスのデータ・ディクショナリ・キャッシュからフラッシュされる必要があります。これは、グローバル・ロックによって実行および同期化されます。

データベース・マウント・ロック

マウント・ロックは、インスタンスが特定のデータベースをマウントしたかどうかを示します。このロックは、Oracle Parallel Server でのみ使用されます。これは、Parallel Server によって排他モードで使用される唯一の複数インスタンス・ロックです。排他モードは、別のインスタンスが共有モードでデータベースをマウントするのを回避します。

Oracle Parallel Server 単一共有モードでは、このロックは共有モードで保持されます。別のインスタンスは、共有モードで同じデータベースを正常にマウントできます。ただし、Parallel Server 排他モードでは、別のインスタンスはロックを取得できません。

パラレル・キャッシュ管理の調整

Oracle Parallel Server がインスタンス間でキャッシュを同期化する方法を理解すると、システム・パフォーマンスに影響を及ぼすオーバーヘッドを理解するのに有効です。5つのノードの Parallel Server があり、ユーザーが1つのノードで表を削除した場合を考えてみます。5つのディクショナリ・キャッシュには、それぞれ削除した表の定義のコピーがあるため、表をキャッシュから削除するノードは、他の4つのディクショナリ・キャッシュからも削除した表の各コピーを削除する必要があります。Oracle Parallel Server は、分散ロック・マネージャを介して、これを自動的に処理します。他のノードのユーザーには、ロック状態で変更が通知されます。

各ノードにライブラリおよび表の情報を格納させることには、重要なメリットがあります。DROP TABLE 文によって他のキャッシュを強制的にフラッシュすることがありますが、これがパフォーマンスに及ぼす影響は、必ずしも、複数キャッシュを持つことのメリットを減少させるわけではありません。

Oracle 内で同期化が必要な処理には、次のものがあります。

- ブロック・レベル・ロック
- 行レベル・ロック
- 領域管理
- システム変更番号

Oracle Parallel Server の排他モードでは、すべての同期化がインスタンス内で行われます。共有モードでは、同期化は、グローバル・ロックの状態を保持する分散ロック・マネージャのコンポーネントの協力によって実現されます。

最も頻繁に要求されるデータベース・リソースは、データ・ブロックです。パラレル・キャッシュ管理によって、1つ以上のタイプのデータ・ブロックを処理するためのグローバル・ロックが提供されます。処理されるタイプは、次のとおりです。

- データ・ブロック
- 索引ブロック
- UNDO ブロック
- セグメント・ヘッダー

パラレル・キャッシュ管理ロックは、インスタンスがデータベース・ブロックを変更または読み取る前にロックを取得するように要求することによって、キャッシュ一貫性を保証します。したがって、パラレル・キャッシュ管理ロックでは、1回に1つのインスタンスのみがブロックを変更できます。

別々のノードに置かれたバッファ・キャッシュのパラレル・キャッシュ管理によって、Parallel Server キャッシュの一貫性が提供されます。パラレル・キャッシュ管理のバッファ・キャッシュ・ロックに対応付けられた一連のグローバル定数 (GC_*) 初期化パラメータは、ディクショナリ・キャッシュやライブラリ・キャッシュなどで使用されるものと同じロックではありません。

パラレル・キャッシュ管理によって、マスター・コピー・データ・ブロック（「一貫読み込みブロック」(すべての変更を保持するバージョンのブロック) ともいう) が変更されようとする場合に、クラスタ内のシステム・グローバル領域の少なくとも1つに保持されることが保証されます。インスタンスがそれを読み込む必要がある場合、カレント・バージョンのブロックは、共有ロックの下にある多くのバッファ・キャッシュに常駐する場合があります。したがって、すべてのシステム・グローバル領域にあるブロックの最新コピーには、インスタンス上のトランザクションがコミットされたかどうかにかかわらず、すべてのインスタンスによるブロックへのすべての変更が含まれます。

1つのバッファ・キャッシュ内でデータ・ブロックが変更された場合、他のバッファ・キャッシュ内のコピーはカレントではなくなります。変更操作が完了すると、新しいコピーを取得できます。

データ・ブロックのカレント・バージョンが1つのインスタンスのバッファ・キャッシュ内にあり、別のインスタンスがそのブロックの更新を要求している場合にのみ、パラレル・キャッシュ管理ロック操作が実行されます。

Oracle Parallel Server の単一のインスタンス上で実行している複数のトランザクションは、グローバル・ロック操作を追加することなく、読み込みに対するデータ・ブロックの集合へのアクセスを共有できます。

この場合、競合はありません。この状態は他のインスタンスで実行しているトランザクションがブロックへの書込みを必要としない限り続きます。

インスタンスは、グローバル・ロックを使用して、リソース・マスター・コピーのオーナーシップを示します。インスタンスがデータベース・リソース・マスター（所有者）になった場合、固定ロックを持つリソースを処理するグローバル・ロックの所有者にもなります。ただし、解放可能なロックは、解放されます。

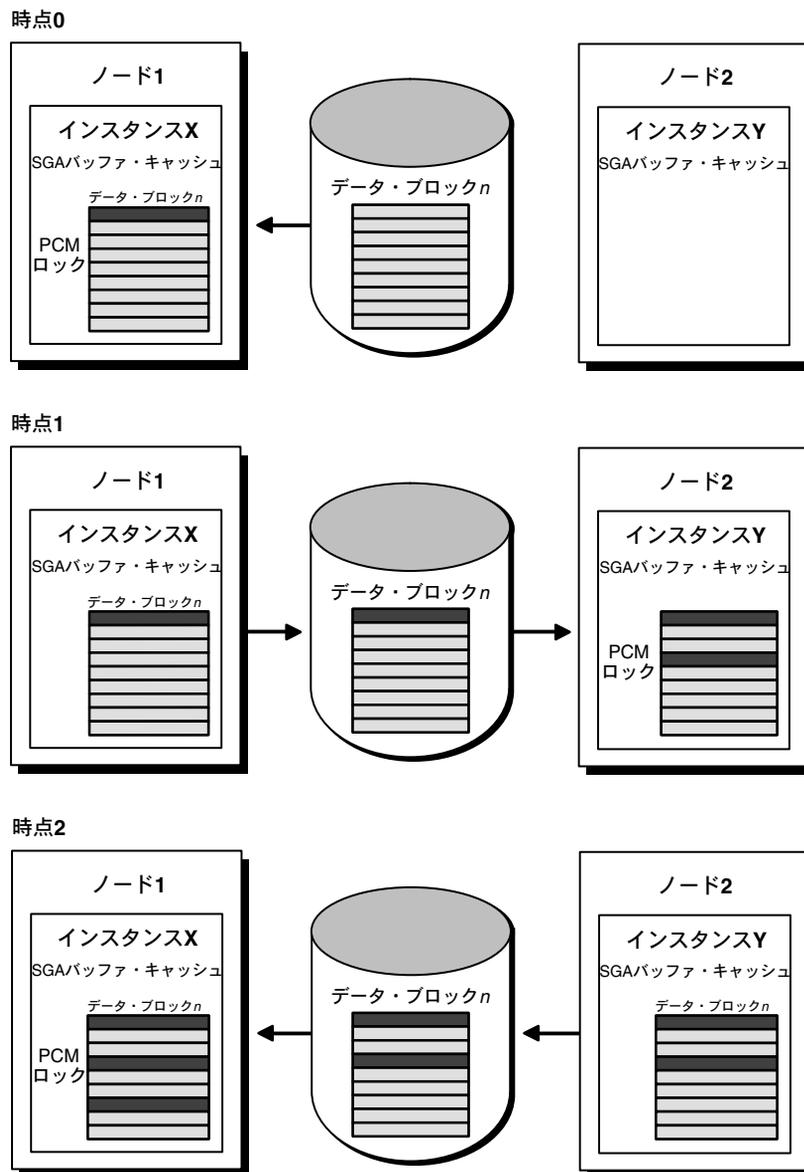
マスター・コピーは、それがリソースの更新可能なコピーであることを示します。インスタンスは、別のインスタンスがリソースの更新を要求した場合にのみ、グローバル・ロックをダウングレードします。一度別のインスタンスがリソースのマスター・コピーを所有すると、そのインスタンスはグローバル・ロックの所有者になります。

パラレル・キャッシュ管理処理の例

次の例および図 4-2 について考えてみます。この例では、1つのパラレル・キャッシュ管理ロックが多くのブロックを処理できる場合でも、1つのブロックを処理すると想定していません。

1. 「時点 0」で、インスタンス X は、行 1 を含むデータ・ブロック n を処理するパラレル・キャッシュ管理ロックの所有者になり、行を更新します。
2. インスタンス Y は、行 4 を更新するためにブロックを要求します。
3. 「時点 1」で、インスタンス X は、データ・ブロックをディスクに書き込み、パラレル・キャッシュ管理ロックを解放します。
4. インスタンス Y は、ブロックおよびパラレル・キャッシュ管理ロックの所有者となり、行 4 を更新します。
5. 「時点 2」で、インスタンス X は行 7 を更新するためにブロックを要求します。
6. インスタンス Y は、データ・ブロックをディスクに書き込み、ブロックおよびパラレル・キャッシュ管理ロックを解放します。
7. インスタンス X は、ブロックおよびパラレル・キャッシュ管理ロックの所有者となり、行 7 を更新します。
8. インスタンス X は、トランザクションをコミットし、別のインスタンスがブロックを要求するまで、パラレル・キャッシュ管理ロックおよびブロックのマスター・コピーを所有します。

図 4-2 複数のインスタンスによる同じデータ・ブロックの更新



パラレル・キャッシュ管理ロックおよび行ロックの独立性

パラレル・キャッシュ管理ロックおよび行ロックは、独立して操作されます。インスタンスは、パラレル・キャッシュ管理ロックによって管理される複数のブロックの集合に保持されている行ロックに影響を及ぼすことなく、パラレル・キャッシュ管理ロックを解放できます。行ロックは、トランザクション中に取得されます。データ・ブロックなどのデータベース・リソースは、インスタンスによって更新のために読み込まれたとき、パラレル・キャッシュ管理ロックを取得します。トランザクション中、他のインスタンスがブロックを必要とした場合に、パラレル・キャッシュ管理ロックは何度も解放および所有されます。

一方、トランザクションは、行に対する変更がコミットまたはロールバックされない限り、行ロックを解放しません。Oracle は、同時実行性を制御する内部メカニズムを使用し、トランザクションを孤立させます。これによって、データを変更しているトランザクションがコミットされるまでは、1つのトランザクションによるデータの変更が、他のトランザクションには参照できないようになります。行ロックの同時実行性制御メカニズムは、パラレル・キャッシュ管理から独立しています。つまり、同時実行性制御はパラレル・キャッシュ管理ロックは必要なく、パラレル・キャッシュ管理ロック操作は、コミットまたはロールバックしている個々のトランザクションに依存しません。

グローバル・ロック・モード

要求されたアクセスのタイプによって、インスタンスは、共有モードまたは排他モードのいずれかで、データ・ブロックの集合を処理するグローバル・ロックを取得できます。

- 排他ロック・モードでは、インスタンスはブロックを更新できます。

あるインスタンスがデータ・ブロックを更新する必要があり、別のインスタンスがそのブロックを処理するグローバル・ロックをすでに所有している場合、最初のインスタンスは、分散ロック・マネージャを使用して、別のインスタンスがグローバル・ロックを解放するように要求します。別のインスタンスは、必要に応じてブロックをディスクに書き込みます。

- 読みみ（共有）ロック・モードでは、インスタンスはブロックの読みみのみ許可されません。

複数インスタンスは、グローバル・ロックによって処理されるブロックを読み込むだけで変更しない限り、共用モードでグローバル・ロックを所有できます。そのため、すべてのインスタンスでは、メモリーに常駐しているブロックのコピーがカレント・コピーであることが保証され、別のインスタンスからブロックを要求するためにグローバル・ロック操作を行わなくてもディスクからカレント・コピーを読み込むことができることが保証されます。これは、読みみ専用の使用目的でアクセスされたデータベースの部分に対してグローバル・ロックを解放する必要がないことを意味します。多くのアプリケーションでは、読みみ専用アクセスが処理時間の大部分を占めています。

- NULL ロック・モードでは、インスタンスはブロックに対する許可なしでロックを保持できます。

このモードは、ロックが継続的に取得および解放される必要がないように使用されます。ロックは、1つのモードから別のモードへ単純に変換されます。

ブロック・レベルのロック

インスタンス間のブロック・アクセスは、ブロックごとのレベルで行われます。インスタンスが排他モードでブロックをロックした場合、他のインスタンスはそのブロックにアクセスできません。Oracle は、データベースからブロックを読み込もうとするたびに、グローバル・ロックを取得する必要があります。したがって、ロックのオーナーシップはインスタンスに割り当てられます。

Oracle Parallel Server は、複数のメモリーのある環境で実行されるため、各インスタンスのメモリーに同じデータ・ブロックの複数のコピーが作成されます。分散ロック・マネージャを使用してノード間の同期化を行うと、ブロックのすべてのコピーの妥当性が保証されます。これらのブロック・レベルのロックは、バッファ・キャッシュ・ロックです。

ブロック・レベルのロックは、Parallel Server が使用可能な場合にのみ発生します。これは、ユーザーおよびアプリケーションに対して透過的です。行レベルのロックは、Parallel Server が使用可能か使用不可かにかかわらず実行されます。

パラレル・キャッシュ管理ロックと非パラレル・キャッシュ管理ロックの比較

パラレル・キャッシュ管理ロックの数は、通常、非パラレル・キャッシュ管理ロックよりも多いです。ただし、分散ロック・マネージャが適切に機能するよう注意して計画する必要のある非パラレル・キャッシュ管理ロックも多くあります。通常、ロックの5～10%は、非パラレル・キャッシュ管理です。非パラレル・キャッシュ管理ロックは、パラレル・キャッシュ管理ロックのように量は増加しません。

パラレル・キャッシュ管理ロックは、初期化パラメータに必要な数を設定することによって、詳細に制御できます。ただし、非パラレル・キャッシュ管理ロックに対しては、ほとんど制御することはできません。DML_LOCKS=0を設定するか、またはALTER TABLE ENABLE/DISABLE TABLE LOCK コマンドを使用することによって、表ロックを排除することができますが、その他の非パラレル・キャッシュ管理ロックは、依然として存続します。

参照： 分散ロック・マネージャのロックに対するリソースの割当てについては、『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を参照してください。

パラレル・キャッシュ管理

この章では、Oracle Parallel Server の分散ロック・マネージャがデータへのアクセスを制御する方法について説明します。分散ロック・マネージャは、[第4章](#)で説明するとおり、他の同期化作業も実行します。データ・アクセスを管理する全体的なプロセス、およびインスタンス間の調整を「パラレル・キャッシュ管理」といいます。この章の内容は次のとおりです。

- [パラレル・キャッシュ管理およびロック実装](#)
- [ロック期間および細分性](#)
- [分散ロック・マネージャによるロック・メカニズムの調整](#)
- [分散ロック・マネージャ・ロックとグローバル・ロックの関連](#)
- [ロック要素およびパラレル・キャッシュ管理ロック](#)
- [パラレル・キャッシュ管理ロックの動作方法](#)
- [パラレル・キャッシュ管理ロックごとのブロック数](#)
- [DLM によるリソース・ロック要求の許可および調整方法](#)
- [ロックの割当ておよび期間の指定](#)

パラレル・キャッシュ管理およびロック実装

Oracle のパラレル・キャッシュ管理機能は、ロックを使用して、複数インスタンスによる共有データのアクセスを調整します。これらのロックを「パラレル・キャッシュ管理ロック」といいます。第 4 章で説明するとおり、Oracle Parallel Server は、非パラレル・キャッシュ管理ロックを使用することによって、データ・ファイルおよび制御ファイルへのアクセスを制御します。

パラレル・キャッシュ管理ロックの数は、非パラレル・キャッシュ管理ロックよりも多くなります。パラレル・キャッシュ管理ロックは、クラスタ内のノードが操作するデータ・ブロックへのアクセスを制御するため、パフォーマンスへの影響がより出やすくなります。そのため、最適のパフォーマンス結果を得るには、パラレル・キャッシュ管理ロックを正確に割り当てることが重要です。

パラレル・キャッシュ管理ロックは、データ・ブロック、索引ブロック、UNDO ブロック、セグメント・ヘッダーなどの、任意のクラスのブロックを 1 つ以上管理できます。ただし、指定されたパラレル・キャッシュ管理ロックは、1 つのブロック・クラスしか処理できません。

パラレル・キャッシュ管理は、要求インスタンスに、ブロックを変更または読み込む前に、データ・ブロックのロックを保持するインスタンスからロックを強制的に取得させることによって、キャッシュ一貫性を保証します。パラレル・キャッシュ管理では、1 回に 1 つのインスタンスのみでブロックの変更が許可されます。あるインスタンスがブロックを変更した場合、別のインスタンスがパラレル・キャッシュ管理ロックを取得してそのブロックを変更する前に、まず、そのブロックはディスクに書き込まれる必要があります。

パラレル・キャッシュ管理ロックは、最小の通信量で、キャッシュ一貫性を保証します。インスタンス間アクティビティの量、および対応する Oracle Parallel Server のパフォーマンスは、ping という観点から評価されます。ping は、1 つのインスタンスが、別のインスタンスに保持されているブロックを要求した場合に発生します。このタイプの要求を解決するために、Oracle は、ブロックをディスクに書き込みまたは ping して、要求インスタンスがブロックを最新の状態で読み込めるようにします。

負荷が大きいアプリケーションでは、かなりのロック・アクティビティが発生する可能性があります。必ずしも過度の ping が発生するわけではありません。データが適切にパーティション化されている場合、ロックは各ノードに対してローカルであるため、ping は発生しません。

キャッシュ一貫性矛盾の解決におけるキャッシュ・フュージョンの役割

データ・ブロックのインスタンス間での参照、およびその結果生じるキャッシュ一貫性の問題は、Oracle Parallel Server のパフォーマンスの主要な問題です。ほとんどの場合、適切にパーティション化することで競合の問題は解決できます。

ただし、実際は、ほとんどのアプリケーションは効果的にパーティション化されていないか、または一部しかパーティション化されていません。インスタンス間の同時アクセスには、次の 3 つのタイプがあります。

- 読み込み / 読み込み
- 読み込み / 書き込み
- 書き込み / 書き込み

読み込み / 読み込みの同時実行性は、2つのインスタンスが同じデータ・ブロックを読み込む必要がある場合に発生します。Oracle Parallel Server では、キャッシュ一貫性の矛盾なしで複数のインスタンスが同じブロックを読み込むために共有できるため、このタイプの競合は簡単に解決できます。ただし、他のタイプの競合は、キャッシュ一貫性の観点から見ると、より複雑です。

読み込み / 書き込みの競合は、多くの場合、OLTP およびハイブリッド・アプリケーションで発生する主な同時実行性の形式です。典型的な適用として意思決定支援システム (DSS) とオンライン・トランザクション処理 (OLTP) を統合できるかは、そのような競合を解決する Oracle Parallel Server の効率に依存します。

Oracle8i では、複数インスタンスによるキャッシュ上でのデータ・ブロックへの同時書き込みアクセスは、ディスク・ベースの ping プロトコルによって管理されます。ここでは、キャッシュ上でのデータ・ブロックに対する現行の変更が、別のインスタンスがそのブロックを読み込みまたは変更する前に、ディスクに書き込まれる必要があります。ディスクへの書き込みは、グローバル・ロック・メカニズムによってブロックへの排他アクセスが付与される前に、行われる必要があります。

Oracle8i では、キャッシュ・フュージョンによって、インターコネクトを使用してインスタンス間で直接データ・ブロックを転送することにより、読み込み / 書き込みの同時実行性を最適化します。これによって、I/O を排除し、ブロック読み込みの待ち時間をプロセス間通信 (IPC) およびインターコネクト・ネットワークのスピードまで減少させます。また、データ・パーティション化の厳しい要件も緩和され、ほとんどの OLTP およびレポート・モジュールなどのバッチ処理で、アプリケーションをより効率的に配置できます。

ロック期間および細分性

ロック期間とは、ロックがリソースに対応付けられている時間の長さを示します。ロックの細分性とは、データ・ブロックあたりのロックの割合を示します。

2つのタイプのロック期間

Oracle 並列・キャッシュ管理では、次の2つのタイプのロック期間を実装します。

- **固定ロック**
- **解放可能ロック**

固定ロック

固定並列・キャッシュ管理ロックは、まず、NULL モードで取得されます。指定されたすべての固定ロックは、インスタンスの起動時に割り当てられ、インスタンス停止時に割当て解除されます。このため、固定ロックは、解放可能なロックよりオーバーヘッドが多く、起動時間が長くなります。ただし、固定並列・キャッシュ管理ロックには、ロックを継続的に取得および解放する必要がないという利点があります。

固定ロックは事前に割り当てられ、起動時にブロックへ静的にハッシュされます。最初に起動するインスタンスは、各固定並列・キャッシュ管理ロックのリソースに、分散ロック・マネージャ・リソースおよび分散ロック・マネージャ・ロックを NULL モードで作成します。その後、これらのロックのモードを必要に応じて他のモードに変換します。後続の各インスタンスは、起動時に NULL モード・ロックを取得し、必要に応じてロック変換を実行します。

デフォルトでは、固定並列・キャッシュ管理ロックは決して解放されません。各ロックは、最後に要求されたモードのまま保持されます。ロックが別のインスタンスに要求された場合、ロックは NULL モードに変換されます。これらのロックは、インスタンス停止時のみ、割当て解除されます。

解放可能ロック

解放可能ロックは、必要に応じて取得および解放されます。したがって、インスタンスは固定ロックの場合よりもずっと速く起動できます。ユーザーが実際にブロックを要求した場合にのみ、分散ロック・マネージャ・リソースが作成され、分散ロック・マネージャ・ロックが取得されます。一度作成された解放可能ロックは、必要に応じて様々なモードに変換できます。

インスタンスは、通常の操作の間に、解放可能ロック・リソースへの参照を解除できます。別のブロックに対するロックの再使用が要求された場合、ロックは解放されます。これは、指定されたリソースのロックを保持するインスタンスが存在しない場合もあることを意味します。

固定ロックと解放可能ロックの比較 固定ロックの場合、インスタンスは、別のインスタンスがロックを要求するまで、決してパラレル・キャッシュ管理ロックを解放しません。そのため、リソースの競合が比較的少なくなり、システムのグローバル・ロック操作のオーバーヘッドが最小化されます。解放可能ロックの場合、一度ブロックが解放されると、そのブロックのロックは再使用できます。非パラレル・キャッシュ管理ロックは、解放されます。

解放可能パラレル・キャッシュ管理ロックは、固定ロックよりもより動的です。解放可能ロックは、必要な場合にのみ分散ロック・マネージャによって割り当てられます。起動時に、要求されたモードで直接取得されたロック要素が割り当てられます。通常は、共有モードまたは排他モードです。

2つのタイプのロックの細分性

Oracle パラレル・キャッシュ管理では、次の2つのタイプのロックの細分性が実装されます。

- 1:1 ロック
- 1:n ロック

1:1 ロック

1:1 ロックとは、1 ブロックあたり1 ロックであることを意味します。これは、ロックの最小の細分性で、デフォルト値です。1:1 ロックは、データベース・オブジェクトが複数のインスタンスによって頻繁に更新される場合に便利です。利点は次のとおりです。

- 競合は、2つのインスタンスによって同じブロックが必要とされた場合にのみ発生します。
- 要求されたブロックのみが、現在パラレル・キャッシュ管理ロックを排他モードで所有しているインスタンスによって、ディスクに書き込まれます。

1:1 ロックの欠点は、ブロックを読み込むごとにオーバーヘッドが発生し、それに伴いパフォーマンスが影響を受けることです。

1:n ロック

1:n ロックは、1つのロックが、nの値で定義されている複数のデータ・ブロックを処理することを示します。1:n ロックの場合、少しのロックで多くのブロックを処理できるため、ロック操作が減少します。読み込み専用データの場合、パラレル実行などの特定の操作の間、1:n ロックは1:1 ロックよりも実行速度が速くなります。

変更が予想されるノードに基づいてデータをパーティション化する場合、各集合が特定のノードに属する分割ロック集合を実装できます。これによって、ロック操作は大幅に減少します。比較的インスタンス変更の少ない大量のデータの場合にも、1:n ロックは効果があります。データを変更するインスタンスによってロックがすでに保持されている場合、その操作に対してロック・アクティビティは必要ありません。

参照： パラレル・キャッシュ管理ロックの構成の詳細は、『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を参照してください。

ロックのコスト

ロックを効率的に実装するには、ロックの相対的成本を慎重に評価します。大まかには、次のことがいえます。

- ラッチはコストが低い
- ローカル・エンキューはややコストが高い
- グローバル・ロックおよびグローバル・エンキューは、非常にコストが高い

一般的に、グローバル・ロックおよびグローバル・エンキューは、パフォーマンスにおいて、等価の効果があります。Oracle Parallel Server が使用不可の場合、すべてのエンキューはローカルです。Parallel Server が使用可能の場合、ほとんどのエンキューはグローバルです。これらの処理にかかる時間の長さは、数千分の1秒単位で異なります。

100 万分の1秒、1000 分の1秒および10 分の1秒は、無視してもいいように思えます。しかし、表 5-1 の「相対所要時間」の列に示すように、極端に大きい値でロックのコストを想像してみてください。

表 5-1 ロックの関連コストの比較

ロックのクラス	実際の所要時間	相対所要時間
ラッチ	100 万分の1秒	1分
ローカル・エンキュー	1000 分の1秒	1,000分 (16時間)
グローバル・ロック (またはグローバル・エンキュー)	10 分の1秒	100,000分 (69日)

表 5-1 では、ロックの使用を慎重に換算する必要があることを強調するために相対例を示したにすぎません。一般的に、統制のとれないグローバル・ロックの使用を避けることが非常に重要です。

参照： アプリケーションで使用されるパラレル・キャッシュ管理ロックの数を分析するための手順については、『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を参照してください。

分散ロック・マネージャによるロック・メカニズムの調整

分散ロック・マネージャは、Oracle Parallel Server の内部にあるリソース・マネージャです。この項では、分散ロック・マネージャがロック・メカニズムを調整する方法について説明します。内容は次のとおりです。

- [分散ロック・マネージャによるロック情報の記録](#)
- [リソース・アクセス権としてのロック・モード](#)
- [インスタンスによる分散ロック・マネージャ・リソースへのデータベース・リソースのマップ](#)
- [分散ロック・マネージャ・ロックとグローバル・ロックの関連](#)
- [リソースにおけるインスタンスごとの1つのロック](#)

リソース・アクセス権としてのロック・モード

Oracle は、最初に、アクセス権を付与せずにリソースにロックを作成することがあります。後でインスタンスが要求を受けた場合、Oracle はロック・モードを変換してアクセス権を取得します。図 5-1 には、アクセス権のレベル、つまり分散ロック・マネージャで使用可能なロック・モードを示します。表 5-2 に、これらのロック・モードおよびその説明を示します。

図 5-1 分散ロック・マネージャのロック・モード：アクセスのレベル

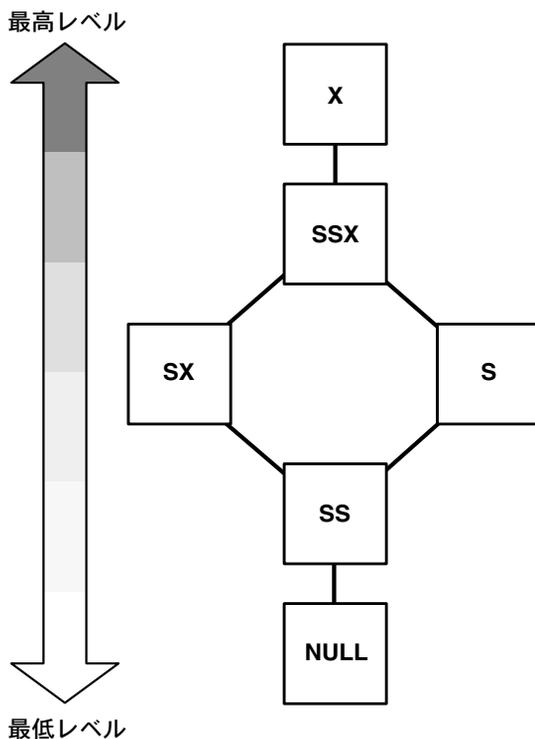


表 5-2 ロック・モード名

ロック・モード	概要	説明
NULL	NULL モード。リソースにロックはありません。	このレベルでロックを保持してもアクセス権はありません。通常、ロックがこのレベルで保持された場合、リソースを必要としているプロセスがあることを示します。または、プレース・ホルダとして使用されます。 NULL ロックが作成されると、リクエストは常にそのリソースのロックを持つことが保証されます。分散ロック・マネージャは、継続してアクセスが必要な場合に、ロックを継続的に作成および破棄する必要はありません。
SS	半共有モード（同時読み込み）。読み込み。書き込み側および他の読み込み側が存在する場合があります。	ロックがこのレベルで保持された場合、対応リソースがプロテクトなしで読み込みます。他のプロセスも対応リソースを読み込みおよび書き込みできます。
SX	共有排他モード（同時書き込み）。書き込み。他の読み込み側および書き込み側が存在する場合があります。	ロックがこのレベルで保持された場合、対応リソースがプロテクトなしで読み込みまたは書き込みできます。他のプロセスも対応リソースを読み込みおよび書き込みできます。
S	共有モード（プロテクト読み込み）。読み込み。書き込み側は許可されません。	ロックがこのレベルで保持された場合、プロセスは対応リソースに書き込みません。複数のプロセスがリソースを読み込みます。これは、従来からの共有ロックです。 共有モードでは、何人のユーザーでも、リソースに同時読み込みアクセスを持つことができます。共有アクセスは読み込み操作に適切です。
SSX	半共有排他モード（プロテクト書き込み）。書き込み側は 1 人のみです。数人の読み込み側が存在する場合があります。	1 つのプロセスのみがこのレベルのロックを保持できます。これによって、1 つのプロセスが、他のプロセスには同時にリソースを変更することを許可しないで、リソースを変更できます。他のプロセスは、プロテクトなしで読み込みができます。従来からの更新ロックです。
X	排他モード。書き込み。他のアクセスは許可されません。	ロックがこのレベルで保持された場合、保持プロセスにリソースへの排他アクセス権が付与されます。他のプロセスはリソースの読み込みも書き込みもできません。これは、従来からの排他ロックです。

インスタンスによる分散ロック・マネージャ・リソースへのデータベース・リソースのマップ

各インスタンスは、Oracle データベース・リソースを分散ロック・マネージャ・リソースへマップします。たとえば、ファイル 2、ブロック 10 などの指定されたデータ・ブロック・アドレスを持つ、Oracle データベース・ブロックの 1:n ロックは、BL 91 など、そのブロックのクラスおよびロック要素番号を持つ「BL リソース」として変換されます。データ・ブロック・アドレスは、Oracle リソース・レベルから分散ロック・マネージャ・リソース・レベルに変換されます。使用されるハッシュ関数は、GC_*パラメータ設定に依存します。

注意： 1:1 ロックの場合、データベース・アドレスは、ロック要素番号としてではなく、2 番目の識別子として使用されます。

図 5-2 リソース名および分散ロック・マネージャのリソース名



分散ロック・マネージャによるロック情報の記録

分散ロック・マネージャは、システム・リソースに対して保持された Oracle グローバル・ロックおよびグローバル・エンキューの一覧表をメンテナンスします。また、競合するロック要求が発生した場合、ネゴシエータとして機能します。この機能を実行する場合、分散ロック・マネージャは、パラレル・キャッシュ管理ロックと非パラレル・キャッシュ管理ロックを区別しません。

サンプル・ロック・マネージャ・ロック・モードおよびリソース一覧表

図 5-3 に、ある Oracle Parallel Server 環境におけるロック・リソース、およびそれらのリソースに対するロックの現行の状態を分散ロック・マネージャの一覧表として示します。

図 5-3 分散ロック・マネージャのリソースおよびロック一覧表の例

統合DLM

リソース	ロック
BL 100, 1	S
BL 101, 1	SSSNNN
BL 3000, 4	X
BL 3001, 4	SSS
BL 100, 6	NNN
BL 101, 6	X
BL 3000, 8	X
BL 3001, 8	N
BL 4000, 9	N

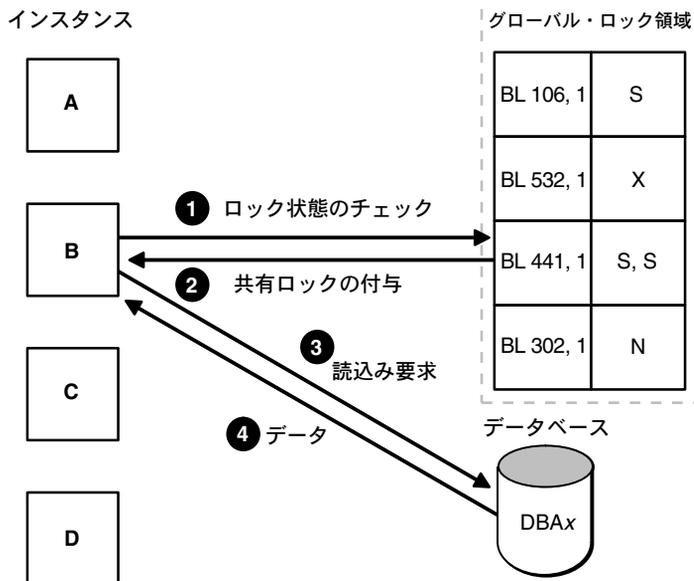
この一覧表の例には、このクラスタ内のすべてのインスタンスが含まれています。たとえば、リソース BL 1,101 は、共有ロックを持つ 3 つのインスタンスと NULL ロックを持つ 3 つのインスタンスに保持されています。図 5-3 では、1 つのリソースで最高 6 つのロックが表示されているため、このシステム上では少なくとも 6 つのインスタンスが実行されています。

分散ロック・マネージャ・ロックとグローバル・ロックの関連

図 5-4 に、分散ロック・マネージャ・ロックとパラレル・キャッシュ管理ロックの関連を示します。インスタンス B がデータ・ブロック・アドレス x のデータ値を読み込むのを許可するには、インスタンス B は、まず、そのアドレスのロックをチェックする必要があります。インスタンス B は、ブロックのデータベース・リソース名を分散ロック・マネージャ・リソース名に変換し、データを読み込むために、分散ロック・マネージャに共有ロックを要求します。

図 5-4 に示されているように、分散ロック・マネージャは、許可キューの未完了のロックをチェックし、リソース 441,BL1 にはすでに 2 つの共有ロックがあることを確認します。共有ロックは、読み専用要求と互換性があるため、分散ロック・マネージャは共有ロックをインスタンス B に許可します。インスタンス B は、データベースに問い合せて、データ・ブロック・アドレス x を読み込みます。データベースはデータを戻します。

図 5-4 分散ロック・マネージャによるロック状態の監視



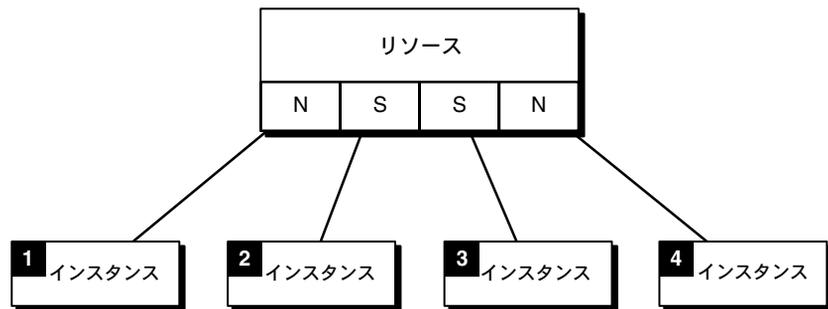
注意: グローバル・ロック領域は、すべてのインスタンスの LMD プロセスによって、分散方式で共同で管理されます。

要求されたブロックにすでに別のインスタンスからの排他ロックがある場合、インスタンス B はそれが解放されるまで待つ必要があります。分散ロック・マネージャは、インスタンス B からの共有ロック要求を変換キューに置きます。分散ロック・マネージャは、排他ロックが削除されたときにインスタンス B に通知し、共有ロックの要求を許可します。

リソースにおけるインスタンスごとの1つのロック

Oracle では、1つのパラレル・キャッシュ管理リソースで、1つのインスタンスにつき1つのロックのみが使用されます。LCK0 プロセスは、リソースへのこのロックの割当てを管理します。図 5-5 に示すように、4つのインスタンスのシステムがあり、単一リソースにバッファ・ロックが必要な場合、実際には、4つのロックが必要です（1インスタンスごとに1つ）。

図 5-5 1 インスタンスにつき1つのロックを持つリソース



非パラレル・キャッシュ管理リソースのロック数は、リソースのタイプ、アプリケーションの動作および構成によって異なります。

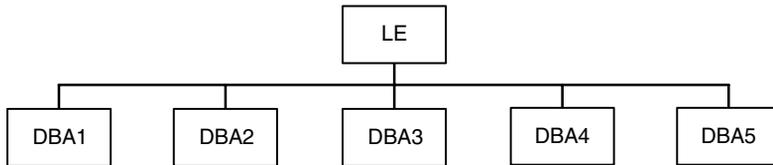
参照： 非パラレル・キャッシュ管理ロックの詳細は、[第 4 章](#)を参照してください。

ロック要素およびパラレル・キャッシュ管理ロック

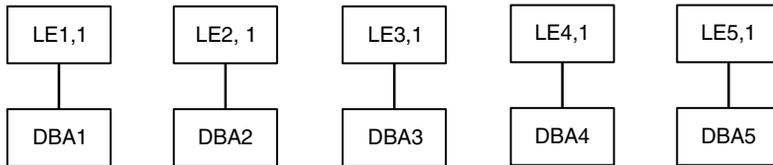
図 5-6 に、固定ロックおよび解放可能ロックのブロックに対応するロック要素を示します。ロック要素 (LE) は、パラレル・キャッシュ管理ロックを表す、Oracle 固有のデータ構造体です。分散ロック・マネージャでは、ロック要素とパラレル・キャッシュ管理ロックは 1 対 1 で対応しています。

図 5-6 1:n ロックおよび 1:1 ロック

1:n ロック (1ロックにつき2つ以上のブロック)



1:1 ロック (1ロックにつき1つのブロック)



固定パラレル・キャッシュ管理ロックのロック要素

固定パラレル・キャッシュ管理ロックおよび解放可能ロックは両方とも、1 ロック要素あたり 2 つ以上のブロックを指定できます。両者の違いは、デフォルトでは、固定パラレル・キャッシュ管理ロックは、解放可能ではない、つまり、ロック要素名が固定されているということです。

リモート要求によってロック要素が ping された場合、そのロック要素が所有している別の変更済ブロックが、要求されたブロックとともに書き込まれます。たとえば、図 5-6 では、ブロック DBA2 (データ・ブロック・アドレス 2) が必要なときに LE が ping された場合、ブロック DBA1、DBA3、DBA4 および DBA5 もすべてディスクに書き込まれます (変更されている場合)。

解放可能パラレル・キャッシュ管理ロックのロック要素

1:1 ロックでは、ロック要素の名前は、分散ロック・マネージャ内のリソースの名前です。一定数のロック要素が潜在的には何万ものブロックを処理できますが、ロック要素名は、要求されるブロックに再度対応付けられるたびに、次々と変更されます。たとえば、ロック要素名 LE7,1 には、データベース・ブロック・アドレス 7、およびロックが処理するブロックのクラス 1 が含まれます。ロック要素を再使用する前に、ロックを解放する必要があります。その後、ロック要素を改名および再使用し、必要に応じて分散ロック・マネージャに新しいリソースを作成できます。

1:1 ロックを使用する場合、ロックは同時に保持する必要がないため、いくつもの潜在的なロック名を使用してシステムを構成できます。ただし、各ロックにマップされたブロック数は、1:n ロックの場合と同じ方法で構成できます。

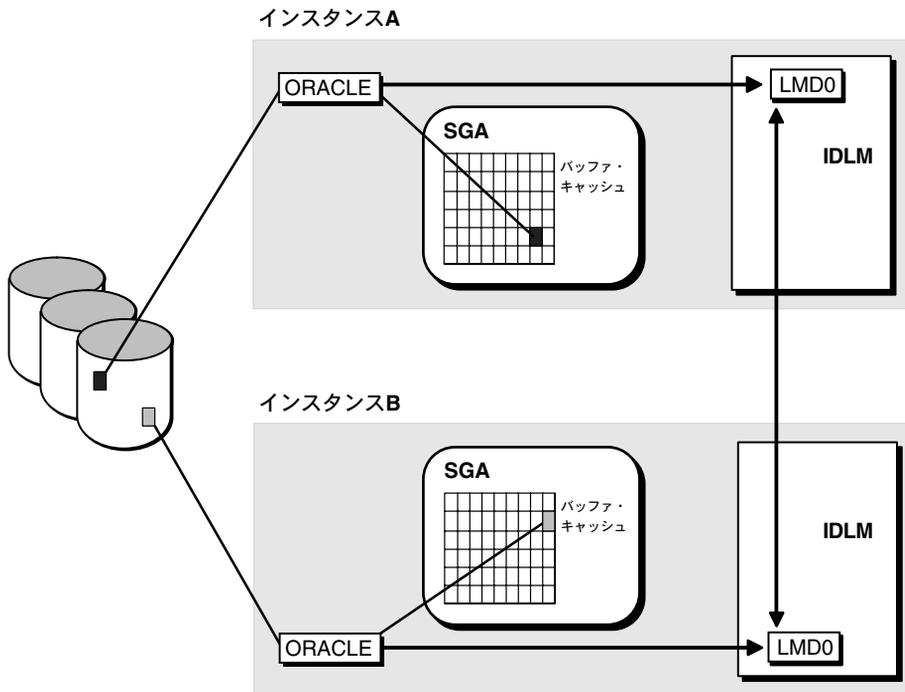
1:1 パラレル・キャッシュ管理ロックのロック要素

1:1 ロックでは、ロック要素とブロックの間に 1 対 1 の関係を設定できます。図 5-6 に示すそのような配置を「データ・ブロック・アドレス・ロック」といいます。したがって、LE2,1 が ping された場合、ブロック DBA2 のみがディスクに書き込まれます。

パラレル・キャッシュ管理ロックの動作方法

図 5-7 に、パラレル・キャッシュ管理ロックがどのように動作するかを示します。インスタンス A が黒のブロックを変更するために読み込む場合、インスタンス A は、そのブロックのパラレル・キャッシュ管理ロックを取得します。これは、網かけのブロックおよびインスタンス B の場合でも同じです。インスタンス B が黒のブロックを要求した場合は、このブロックはインスタンス A によってすでに変更されているため、ディスクに書き込まれる必要があります。Oracle プロセスは LMD プロセスと通信し、分散ロック・マネージャからグローバル・ロックを取得します。

図 5-7 パラレル・キャッシュ管理ロックの動作方法



インスタンス LCK プロセスによって所有されるパラレル・キャッシュ管理ロック

各インスタンスには、少なくとも1つの LCK バックグラウンド・プロセスがあります。同じインスタンス内に複数の LCK プロセスが存在する場合、パラレル・キャッシュ管理ロックは、LCK プロセス間で分割されます。これは、各 LCK プロセスは、パラレル・キャッシュ管理ロックのサブセットに対してのみ責任があることを意味します。

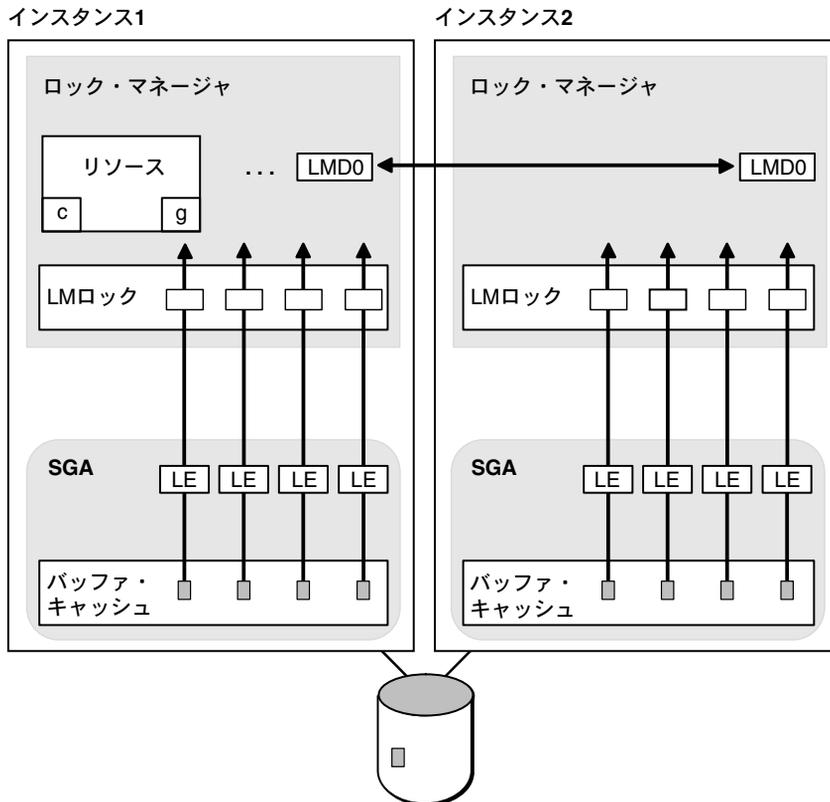
複数インスタンスによる同じロックの所有

共有モードで所有されているパラレル・キャッシュ管理ロックは、別のインスタンスが共有モードでパラレル・キャッシュ管理ロックを要求しても解放されません。したがって、コピーが共有されるため（書込みは発生しません）、2つのインスタンスがそれぞれのバッファ・キャッシュに同じデータ・ブロックを持つことができます。同じパラレル・キャッシュ管理ロックに処理される異なるデータ・ブロックが、別々のインスタンスのバッファ・キャッシュに含まれます。これは、すべての異なるインスタンスが、共有モードでパラレル・キャッシュ管理ロックを要求した場合に起こります。

1:1 ロックの動作方法

図 5-8 に、1:1 ロックがどのように動作するかを示します。

図 5-8 ロック要素によるブロックの調整 (1:1 ロックの場合)



フォアグラウンド・プロセスは、システム・グローバル領域をチェックし、インスタンスがブロックのロックを所有しているかどうかを判断します。

- ロックが所有されていないかまたは存在しない場合、フォアグラウンド・プロセスは、別のロックを解放することによって、固定ロックまたは解放可能ロックを作成します。
- インスタンスがロックを所有していても、それが不適切なモードの場合、フォアグラウンド・プロセスは、たとえば、共有モードから排他モードへロックのモードを変換します。

- 固定ロックでは、1:n か 1:1 かにかかわらず、固定モードでロック要素名が作成されます。これは常に有効です。このモードは静的であるため、ロック要素は割り当てられたままの状態を保持します。
- 解放可能ロックでは、1:n か 1:1 かにかかわらず、解放可能モードでロック要素が作成されます。これらのロック要素名および処理されるブロックは変更できます。非固定モードのロック要素は、有効、古いまたは空きになります。有効ビットが設定されている場合、ロックは分散ロック・マネージャのリソースで所有されます。設定されていない場合、ロックはありません。空きの場合、ロックはありますが、ロック要素からバッファをリンク解除しているため、それは、空きロック要素の LRU リスト上にあります。

注意： 有効なロック要素は分散ロック・マネージャにロックを持っていますが、無効なロック要素は持っていません。空きロック要素は、現在このバッファにリンクされていない分散ロック・マネージャにロックが存在していることを示します。このロックは、LRU リストで待機しています。ロック要素が古い場合は、古い名前についての有効ロック・ハンドルがあります。Oracle がそれを使用できるようにするには、新しい名前を付ける必要があります。

パラレル・キャッシュ管理ロックごとのブロック数

データ・ファイルに割り当てられたパラレル・キャッシュ管理ロックの数、およびそれらのデータ・ファイル内のデータ・ブロックの数によって、1つのパラレル・キャッシュ管理ロックによって処理されるデータ・ブロックの数が決定されます。

- GC_FILES_TO_LOCKS がファイルに設定されていない場合、1:1 の解放可能ロックが各ブロックに1つ使用されます。
- GC_FILES_TO_LOCKS がファイルに設定されている場合、パラレル・キャッシュ管理ロックごとのブロック数は、ファイル・レベルごとに次のように計算できます。この例では、GC_FILES_TO_LOCKS = 1:300,2:200,3-5:100 の値を想定しています。

$$\text{ファイル1: } \frac{\text{ファイル1のブロック数}}{300\text{ロック}}$$

$$\text{ファイル2: } \frac{\text{ファイル2のブロック数}}{200\text{ロック}}$$

$$\text{ファイル3: } \frac{\text{合計 (ファイル3、ファイル4、ファイル5のブロック数)}}{100\text{ロック}}$$

ブロック数によるファイルのサイズが、それに割り当てられたパラレル・キャッシュ管理ロック数の倍数の場合、各 1:n パラレル・キャッシュ管理ロックは、等式で求められるデータ・ブロック数と同数のデータ・ブロックを処理します。

ファイル・サイズが、パラレル・キャッシュ管理ロック数の倍数でない場合、1:n パラレル・キャッシュ管理ロックごとのデータ・ブロック数は、個々のデータ・ファイルに対して1つずつ異なります。たとえば、2,500 個のデータ・ブロックを含むデータ・ファイルに、400 個のパラレル・キャッシュ管理ロックを割り当てた場合、100 個のパラレル・キャッシュ管理ロックは7データ・ブロックずつ処理し、300 個のパラレル・キャッシュ管理ロックが6ブロックずつ処理します。GC_FILES_TO_LOCKS 初期化パラメータに指定されていないすべてのデータ・ファイルは、残りのパラレル・キャッシュ管理ロックを使用します。

n 個のファイルが同じ 1:n パラレル・キャッシュ管理ロックを共有する場合、ロックごとのブロック数は最大で n 個異なります。GC_FILES_TO_LOCKS の分割句またはキーワード EACH を使用して、個々のファイルにロックを割り当てる場合、ロックごとのブロック数の違いは 1 以下になります。

1:n パラレル・キャッシュ管理ロックをデータ・ファイルの集合にまとめて割り当てた場合、各ロックは通常、各ファイルの1つ以上のブロックを処理します。連続するブロックを指定した場合 (*!blocks* オプションを使用して)、またはファイルに含まれるブロック数が、ファイルの集合に割り当てたロック数より少ない場合には、例外が発生する場合があります。

複数ブロックを処理するロックの例

次に、1:n パラレル・キャッシュ管理ロックが異なるファイル内の複数ブロックを処理する方法を示します。図 5-9 では、合計 44 個のブロックがある 2 つのファイルに割り当てられた 44 個のパラレル・キャッシュ管理ロックを想定しています。GC_FILES_TO_LOCKS は、A,B:44 に設定されています。

ファイルのブロック 1 は、ロック 1 から始まるというわけではありません。ファイルがどのロックから始まるかは、ハッシュ関数によって決定されます。24 個のブロックを持つファイル A では、ブロック 1 がロック 32 にハッシュします。20 個のブロックを持つファイル B では、ブロック 1 がロック 28 にハッシュします。

図 5-9 複数ファイルのブロックを処理する固定パラレル・キャッシュ管理ロック

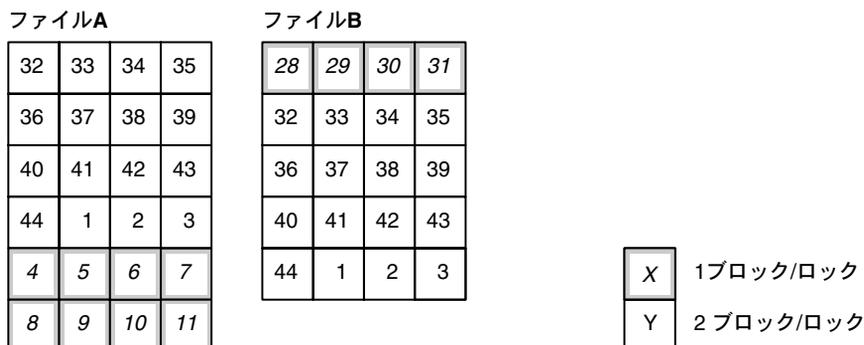


図 5-9 では、ロック 32 ~ 44 および 1 ~ 3 は、それぞれ 2 つのブロックを処理するために使用されます。ロック 4 ~ 11 および 28 ~ 31 は、それぞれ 1 ブロックずつを処理します。また、ロック 12 ~ 27 はどのブロックも処理しません。

最悪のケースとして考えられるのは、2 つのファイルが同じロックを開始点としてハッシュした場合で、すべての共通ロックがそれぞれ 2 つのブロックを処理することになります。ファイルが大きく、ロックごとに複数のブロックがある場合 (1 ロックあたり 100 ブロック程度)、これは重大な問題ではありません。

固定パラレル・キャッシュ管理ロックの周期性

パラレル・キャッシュ管理ロックの周期性についても考慮する必要があります。図 5-10 に、6つのパラレル・キャッシュ管理ロックによって処理された 30 個のブロックを持つファイルを示します。このファイルは、ロック番号 5 で始まるよう設定された 1:n ロックを持っています。パラレル・キャッシュ管理ロック番号 4 で処理された黒いブロックが示すように、各ロックの使用によって、ファイル内にブロックのパターンが形成されます。

図 5-10 固定パラレル・キャッシュ管理ロックの周期性

5	6	1	2	3
4	5	6	1	2
3	4	5	6	1
2	3	4	5	6
1	2	3	4	5
6	1	2	3	4

ping: 更新の必要性の通知

Parallel Server では、特定のデータ・ブロックは、一度に 1 つのインスタンスによってのみ変更できます。あるインスタンスが、別のインスタンスが要求しているデータ・ブロックを変更する場合、ping が必要かどうかはブロックに発行された要求のタイプによります。

要求側インスタンスがブロックを変更しようとしている場合、保持側インスタンスのデータ・ブロックに対するロックは、それに合わせて変換される必要があります。最初のインスタンスは、要求側インスタンスがそのブロックを読み込む前に、ブロックをディスクに書き込む必要があります。これを、ブロックを ping するといいます。

BSP (ブロック・サーバー・プロセス) は、分散ロック・マネージャの機能を使用して、2 つのインスタンス間で必要性を通知します。要求側インスタンスが CR モードのブロックのみを必要としている場合、保持側インスタンスの BSP は、インターコネクトを介して、ブロックの CR バージョンを要求側インスタンスへ送信します。この場合、ping はかなり高速に実行されます。

データ・ブロックは、別のインスタンスが、1 つのインスタンスのバッファ・キャッシュで、排他カレント (XCUR) 状態で保持されたブロックを変更するために必要としている場合にのみ ping されます。そのため、場合によっては、データ・ブロックを処理するパラレル・キャッシュ管理ロックの数は、ブロックが ping されるかどうかにはほとんど影響しません。

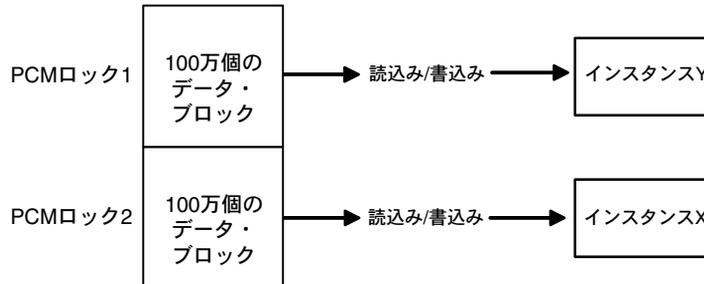
インスタンスは、ブロックの排他ロックを破棄しても、その中の行の行ロックは保持し続けることができます。ping は、コミットが発生したかどうかには関係ありません。ブロックは変更できますが、そのブロックが ping されるかどうかは、コミットしたかどうかには関係ありません。

ping を回避するためのパーティション化

インスタンス間でデータをパーティション化し、更新を行っている場合、アプリケーションは、たとえば、各インスタンスに 100 万個のブロックを持つことができます。各ブロックが 1 つのパラレル・キャッシュ管理ロックによって処理されるとしても、強制的な書込みまたは読み込みは発生しません。

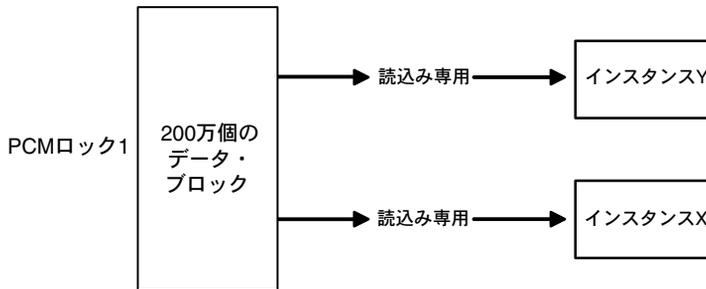
図 5-11 で示すように、単一のパラレル・キャッシュ管理ロックが表内の 100 万個のデータ・ブロックを処理し、その表のブロックがインスタンス X のシステム・グローバル領域へ読み込まれたり書き込まれたりすると想定します。また、別の単一のパラレル・キャッシュ管理ロックが同じ表内の別の 100 万個のデータ・ブロックを処理し、それらがインスタンス Y のシステム・グローバル領域へ読み込まれたり書き込まれたりすると想定します。更新の回数にかかわらず、インスタンス X とインスタンス Y の間でデータ・ブロックへの強制読み込みまたは強制書込みは発生しません。

図 5-11 ping を回避するためのデータのパーティション化



読み専用データでは、インスタンス X およびインスタンス Y の両方が、ping を発生させることなく、共有モードでパラレル・キャッシュ管理ロックを保持できます。図 5-12 に、この例を示します。

図 5-12 ping が発生しない読み専用データ



参照： ping を回避するためのアプリケーションのパーティション化については、『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を参照してください。

ロック・モードおよびバッファ状態

バッファ・キャッシュ内のブロックの状態は、それに保持されているロックのモードに直接関係します。たとえば、バッファが排他カレント (XCUR) 状態の場合、インスタンスが排他モードでパラレル・キャッシュ管理ロックを所有することがわかります。データベースには、1つの XCUR バージョンのブロックのみがある場合も、複数の SCUR バージョンがある場合もあります。変更を実行するには、プロセスはブロックを XCUR モードで取得する必要があります。

バッファの状態の検索

バッファの状態を確認するには、V\$BH 動的パフォーマンス表の STATUS 列をチェックします。この表には、各バッファ・ヘッダーについての情報があります。

表 5-3 パラレル・キャッシュ管理ロック・モードおよびバッファの状態

パラレル・キャッシュ管理ロック・モード	バッファ状態名	説明
X	XCUR	インスタンスは、このバッファについて排他ロックを所有している
S	SCUR	インスタンスは、このバッファについて共有ロックを所有している
N	CR	インスタンスは、このバッファについて NULL ロックを所有している

バッファ状態およびロック・モードの変化

図 5-13 に、インスタンスが指定されたバッファに対して様々な操作を実行するに従って、バッファの状態およびロック・モードがどのように変化するかを示します。ロック・モードはカッコで囲んで示しています。

図 5-13 バッファ状態およびロック・モードの変化

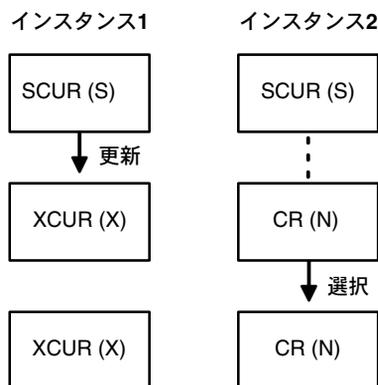


図 5-13 では、3つのインスタンスが共有カレント・モードのブロック、および共有ロックで起動します。インスタンス 1 がブロックの更新を実行する場合、ブロックのロック・モードは排他モード (X) に変更されます。インスタンス 2 およびインスタンス 3 に所有される共有ロックは、NULL モード (N) に変換されます。そのとき、インスタンス 1 のブロック状態は XCUR になり、インスタンス 2 およびインスタンス 3 では CR になります。これらのロック・モードには互換性があります。

互換または非互換のロック・モード

あるプロセスが特定のモードでロックを所有している場合、ある特定のモードでロックを要求している別のプロセスは、表 5-4 に示すように成功または失敗します。

表 5-4 ロック・モードの互換性

要求されたロック：	NULL	SS	SX	S	SSX	X
所有しているロック						
NULL	成功	成功	成功	成功	成功	成功
SS	成功	成功	成功	成功	成功	失敗
SX	成功	成功	成功	失敗	失敗	失敗
S	成功	成功	失敗	成功	失敗	失敗
SSX	成功	成功	失敗	失敗	失敗	失敗
X	成功	失敗	失敗	失敗	失敗	失敗

DLM によるリソース・ロック要求の許可および調整方法

この項では、分散ロック・マネージャがリソース・ロック要求を調整する方法について説明します。内容は次のとおりです。

- [ロック要求のキュー](#)
- [非同期トラップ \(AST\) によるロック要求状態の通知](#)
- [ロック要求の変換および許可](#)

分散ロック・マネージャは、すべてのロック要求を追跡し、可能な場合にはリソースに対する要求を許可します。現在使用不可のリソースへの要求も追跡し、それらのリソースが後に使用可能になったときに、アクセス権を付与します。分散ロック・マネージャは、ロック要求を管理し、ユーザー、およびパラレル・キャッシュ管理に関する内部プロセスにそれらの状態を通信します。

ロック要求のキュー

分散ロック・マネージャは、ロック要求に対する次の2つのキューを保持します。

変換キュー	分散ロック・マネージャがすぐにロック要求を許可できない場合、ロック要求は変換キューに入れられ、そこで待機中のロック要求が追跡されます。
許可キュー	分散ロック・マネージャは、許可キューで許可されたロック要求を追跡します。

非同期トラップ (AST) によるロック要求状態の通知

ロック要求の状態を通知するために、分散ロック・マネージャは2種類の非同期トラップ (AST) または割込みを使用します。

獲得 AST	ロックが要求のモードで取得された場合、獲得 AST (ウェイクアップ・コール) が送信され、ロックが許可されたことがリクエストに通知されます。
ブロッキング AST	プロセスがリソースの特定モードのロックを要求した場合、分散ロック・マネージャはブロッキング AST を送信して、そのリソースのロックを現在非互換モードで所有しているプロセスに通知します。(たとえば、共有モードと排他モードは非互換です。) 通知されると、ロックの所有者はロックを破棄して、リクエストによるアクセスを許可できます。

ロック要求の変換および許可

次の図に、分散ロック・マネージャが要求を処理する方法を示します。図 5-14 では、リソースに対する共有ロック要求 1 がプロセス 1 に、共有ロック要求 2 がプロセス 2 に許可されています。前述のとおり、分散ロック・マネージャは許可キュー内のロックを追跡します。プロセス 2 が排他ロックを要求した場合、その要求は変換キュー内で待機する必要があります。

図 5-14 分散ロック・マネージャの許可キューおよび変換キュー

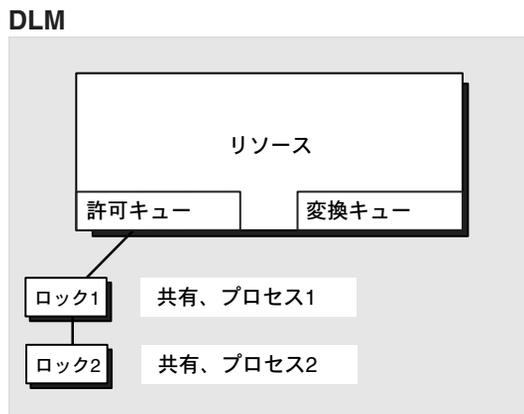
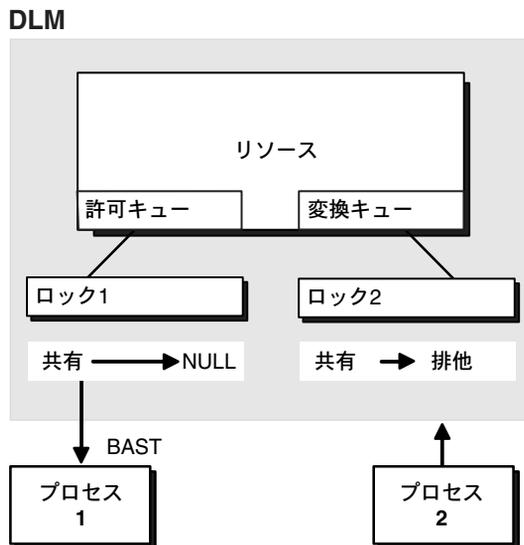


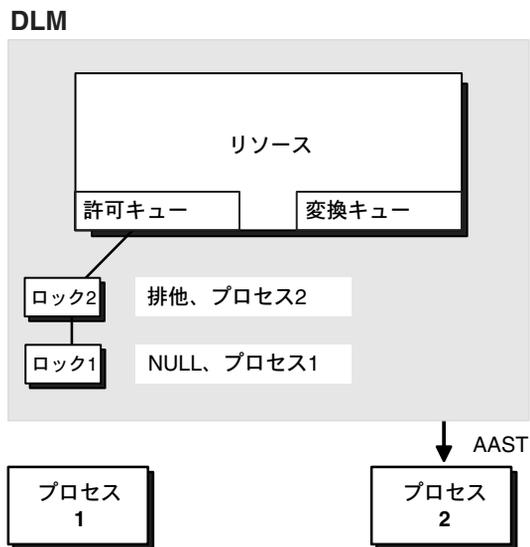
図 5-15 では、分散ロック・マネージャは共有ロックの所有者であるプロセス 1 にブロッキング AST を送信し、排他ロックの要求が待機していることを通知します。共有ロックがプロセス 1 によって破棄されると、ロックは NULL モードに変換されるか、または解放されます。

図 5-15 ブロッキング AST



獲得 AST は、その後、排他ロックのリクエストであるプロセス 2 に警告を送信します。分散ロック・マネージャは、排他ロックを許可し、それを許可キューへ変換します。図 5-16 に、これを示します。

図 5-16 獲得 AST



ロックの割当ておよび期間の指定

パラレル・キャッシュ管理ロックをデータ・ファイルに割り当てるには、パラメータ・ファイルの初期化パラメータに値を指定します。このパラメータ・ファイルは、データベースの起動時に読み込まれます。たとえば、初期化パラメータ `GC_FILES_TO_LOCKS` を使用して、単一のデータ・ファイルまたは複数のデータ・ファイルのデータ・ブロックを処理するパラレル・キャッシュ管理ロックの数を指定します。

パラレル・キャッシュ管理ロックごとのブロック数

この項では、1:n ロックおよび 1:1 ロックでロックの細分性を変更する方法について説明します。

複数ブロックに対する 1:n ロック

ファイル内の連続するブロックの範囲を保護する、ロック対ブロックの割合を指定できます。表 5-5 に、1:n ロックが役立つ状況の概要を示します。

表 5-5 1:n パラレル・キャッシュ管理ロックを使用する場合

状況	理由
データがほとんど読み専用の場合	少数の 1:n ロックで、頻繁にロック・オペレーションを要求することなく多くのブロックを処理できます。これらのロックは、別のインスタンスがデータを変更する必要がある場合にのみ解放されます。パラレル問合せオプションを使用する場合、読み専用データの 1:n ロックは 1:1 ロックよりも高速に実行できます。データが厳密に読み専用の場合、表領域自体を読み専用として設計することを考慮してください。その場合、表領域は、パラレル・キャッシュ管理ロックを必要としません。
データを変更する可能性のあるインスタンスに基づいて、データをパーティション化できる場合	このパーティション化に一致するよう定義された 1:n ロックでは、インスタンスは、分散ロック・マネージャの分割されたロックの集合を保持でき、分散ロック・マネージャの操作の必要性が削減されません。
比較的少量のインスタンス集合によって大量のデータが変更される場合	要求インスタンスによってロックがすでに保持されている場合、1:n ロックでは、分散ロック・マネージャのアクティビティなしで、新しいデータベース・ブロックへのアクセスが許可されます。

1:n ロックを使用すると、別々のデータベース・ブロックを変更するインスタンス間で競合が発生する場合があります。余分なインスタンス間のロック・アクティビティ (false ping) が発生することがあります。false ping を解決するには、現在ロックを所有しているインスタンスのキャッシュから、いくつかのブロックを書き込む必要がある場合があります。GC_FILES_TO_LOCKS パラメータを適切に設定することによって、false ping を最小化または排除できます。

1:1 ロック : 1つのブロックに対するロック

パラレル・キャッシュ管理ロックとデータ・ブロックを1対1で対応付ける場合、インスタンスが同じブロックからデータを必要とする場合にのみ競合が発生します。1:1ロックのこのレベルを「DBA ロック」ともいいます。この場合、DBA は、データ・ブロックのデータ・ブロック・アドレスです。1ロックあたりに2つ以上のブロックを割り当てた場合、1:nロックの場合のように競合が発生します。

ほとんどのシステムでは、システム・グローバル領域または分散ロック・マネージャの使用メモリーが大きくなりすぎてしまうため、インスタンスはデータベースの各ブロックに対してロックを保持できません。そのため、インスタンスは、必要に応じて1:1ロックを取得および解放します。1:1ロック、ロック要素およびリソースは、分散ロック・マネージャで改名されて再使用されるため、システムはそれらを少数だけ使用して適切に機能できます。DB_BLOCK_BUFFERS パラメータに設定する値は、割り当てる必要のある解放可能ロックの推奨最小数です。

ロックの細分性の選択

表 5-6 の情報を参照して、1:n ロックか 1:1 ロックのどちらを使用するかを判断してください。

表 5-6 1:n ロックおよび 1:1 ロックを使用する場合

1:n ロックを使用する場合	1:1 ロックを使用する場合
<ul style="list-style-type: none">■ データがほとんど読み専用の場合	<ul style="list-style-type: none">■ 少量のデータが多くのインスタンスによって更新される場合
<ul style="list-style-type: none">■ データがパーティション化できる場合	<ul style="list-style-type: none">■ 1:n ロックを構成するにはメモリーが十分でない場合
<ul style="list-style-type: none">■ 比較的少量のインスタンス集合によって大量のデータ集合が変更される場合	

固定ロックおよび解放可能ロックの同時使用

表 5-7 に、固定ロックと解放可能ロックを同時に使用する場合の比較を示します。

表 5-7 固定キャッシュ管理ロックおよび解放可能キャッシュ管理ロックの使用

固定パラレル・キャッシュ管理ロック	解放可能パラレル・キャッシュ管理ロック
<ul style="list-style-type: none"> ■ インスタンスの起動時に割り当てられ、起動が遅くなる。 ■ インスタンス停止時にのみ解放される。 ■ 起動時に静的にブロックに割り当てられ、多くのメモリーが必要になる。 	<ul style="list-style-type: none"> ■ ユーザーがブロックを要求した場合に割り当てられ、インスタンスの起動が速くなる。 ■ 他のブロックによって動的に再使用され、多くのメモリーを必要としない。

グループ所有のロック

グループ・ベースのロックでは、動的なオーナーシップが提供されます。単一のロックは、同じグループに属する複数のプロセスによって共有されます。同じグループのプロセスは、新しいロックおよび別のロックをオープンまたは変換しないで、ロックを共有またはアクセス（あるいはその両方）できます。これは、特に、マルチスレッド・サーバーおよびXAに重要です。

マルチスレッド・サーバーおよびXA に対する分散ロック・マネージャのサポート

Oracle Parallel Server では、2つの形式のロック・オーナーシップが使用されます。

プロセスごとのオーナーシップ	ロックは通常、プロセス所有です。つまり、1つのプロセスがロックを所有すると、他のプロセスはどれもそのロックにアクセスできません。
グループ・ベースのオーナーシップ	グループ・ベースのロックの場合、オーナーシップは動的になります。単一のロックを同じグループに属する複数のプロセスが使用できます。同じグループ内のプロセスは、分散ロック・マネージャの許可キューおよび変換キューに送信することなく、ロックを交換またはアクセス（あるいはその両方）できます。

グループ・ベースのロックは、Oracle マルチスレッド・サーバー（MTS）およびXA ライブラリ機能の重要な分散ロック・マネージャ機能です。

MTS	グループ・ベースのロックは、Oracle MTS 構成に使用されます。グループ・ベースのロックがないと、セッションは共有サーバー・プロセス間で移行できません。さらに、特に長時間実行のトランザクションでは、ロード・バランスが影響を受けることがあります。
XA ライブラリ	Oracle XA ライブラリでは、複数のセッションまたはプロセスがトランザクションで動作できます。そのため、それらは、排他モードであっても、同じロックを交換する必要があります。グループ・ベースのロック・オーナーシップでは、プロセスは、排他リソースへのアクセスを交換できません。

分散ロック・マネージャに対するメモリー要件

ユーザー・レベルの分散ロック・マネージャでは、通常、要求しただけのメモリーが割り当てられます。ただし、プロセス・サイズはそれに伴い増加します。これは、ロックおよびリソースがアドレス空間に常駐する共有メモリーをマップしているためです。そのため、プロセス・アドレス空間は非常に大きくなる可能性があります。

分散ロック・マネージャが、アプリケーションに必要なすべてのリソースをサポートするよう構成されていることを確認してください。

第III部

Oracle Parallel Server の実装

第 III 部では、Oracle Parallel Server の実装に固有のトピックについて説明します。第 III 部に含まれる章は、次のとおりです。

- 第 6 章「Oracle Parallel Server コンポーネント」
- 第 7 章「Oracle Parallel Server の記憶域上の考慮事項」
- 第 8 章「スケーラビリティおよび Oracle Parallel Server」
- 第 9 章「高可用性および Oracle Parallel Server」

Oracle Parallel Server コンポーネント

この章では、Oracle Parallel Server を実装するためのコンポーネントについて説明します。
この章の内容は次のとおりです。

- Oracle Parallel Server のインスタンスおよびデータベース・コンポーネント
- キャッシュ・フュージョン処理およびブロック・サーバー・プロセス
- システム変更番号処理

Oracle Parallel Server のインスタンスおよびデータベース・コンポーネント

この項では、プロセスおよびコンポーネントの観点から、Oracle Parallel Server 固有の実装の概念について説明します。各 Oracle Parallel Server インスタンスには、独自のシステム・グローバル領域および単一インスタンスに共通のバックグラウンド・プロセスがあります。各インスタンスには、独自の一連の制御ファイル、データ・ファイルおよび REDO ログも含まれます。また、クラスタ内のインスタンスは、クラスタを通じて、すべてのシステム・グローバル領域、制御ファイル、データ・ファイルおよび REDO ログを共有またはアクセスします。

その他の Parallel Server 固有のコンポーネントは、次の項で説明します。これらのコンポーネントによって、パラレル処理が可能になり、ノード間通信が行われます。

Parallel Server 固有のプロセス

単一インスタンスに共通の PMON、SMON、DBWR などのプロセスに加えて、Oracle Parallel Server インスタンスには、クラスタ内のノード間でのリソース共有を行う別のプロセスがあります。

参照： Oracle プロセスの詳細は、『Oracle8i 概要』を参照してください。

4 つの Parallel Server 固有のプロセスによって、リソース共有が行われます。最初の 3 つのプロセスは、分散ロック・マネージャ・コンポーネントとともに動作して、グローバル・ロックおよびリソースを管理します。

- LMON – 「ロック・モニター・プロセス」は、クラスタ全体を監視してグローバル・ロックおよびリソースを管理します。LMON は、インスタンスおよびプロセスの異常終了、および分散ロック・マネージャに関連するリカバリを管理します。特に、LMON はグローバル・ロックに関連するリカバリの部分を処理します。
- LMD – 「ロック・マネージャ・デーモン」は、ロック・エージェント・プロセスです。これは、パラレル・キャッシュ管理ロックに対するロック・マネージャ・サービス要求を管理し、グローバル・ロックおよびリソースへのアクセスを制御します。LMD プロセスは、デッドロック検出およびリモート・ロック要求も処理します。リモート・ロック要求とは、別のインスタンスから発生する要求のことです。
- LCK – 「ロック・プロセス」は、非パラレル・キャッシュ管理ロックを管理して、共有リソースおよびそれらのリソースへのリモート要求を調整します。
- BSP – 「ブロック・サーバー・プロセス」は、インスタンス間の読込み / 書込み要求によって要求されたブロックに対するコミットされていないトランザクションをロールバックし、一貫読込みブロックをリクエストに送信します。

インスタンスが起動すると、LMON および LMD プロセスが起動され、分散ロック・マネージャが Cluster Manager に登録されます。データベースを停止すると、分散ロック・マネージャは Cluster Manager から登録解除されます。

共有モードでインスタンスに障害が発生した場合、別のインスタンスの SMON が障害を検出し、障害インスタンスをリカバリします。リカバリを実行しているインスタンスの LMON プロセスは、障害インスタンスの PCM ロックを再マスターします。

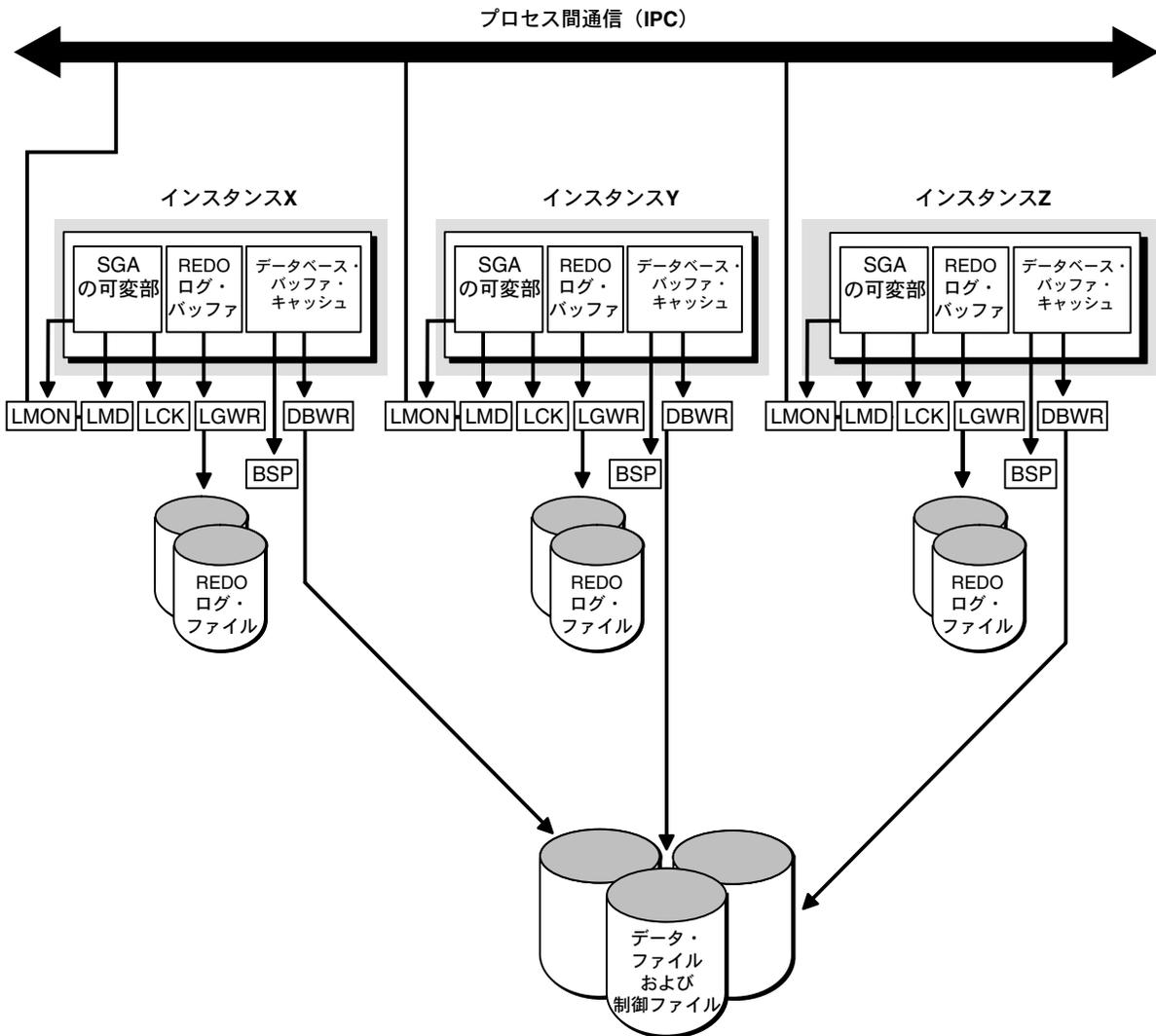
フォアグラウンド・プロセスおよびフォアグラウンド・ロック取得

フォアグラウンド・プロセスとは、ユーザー・セッションに対応付けられたプロセスのことです。1つのインスタンスは、ロック要求を直接他のインスタンスのリモート LMD プロセスに通知します。フォアグラウンド・プロセスは、ロックを要求しているリソース名、ロックが必要な要求のモードなどの要求情報を送信します。分散ロック・マネージャは、非同期に要求を処理するため、フォアグラウンド・プロセスは、要求をクローズする前に、要求の完了を待ちます。

Oracle Parallel Server プロセスの概要

図 6-1 に、Oracle Parallel Server 固有のプロセス、および Oracle が単一インスタンス環境でも使用するいくつかのプロセスを示します。

図 6-1 Oracle Parallel Server の基本要素



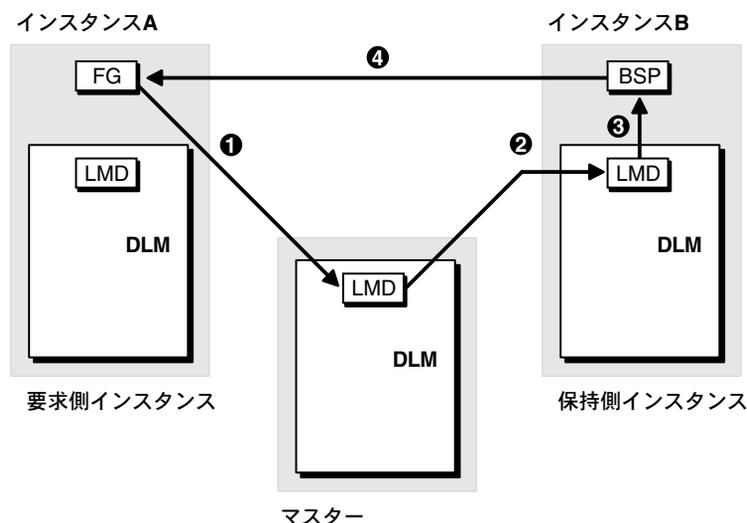
キャッシュ・フュージョン処理およびブロック・サーバー・プロセス

1つのインスタンスが、別のインスタンスによって排他モードで保持されているブロックを要求する場合、キャッシュ・フュージョンによってキャッシュの一貫性競合が解決されます。このような場合、一貫読みバージョンのブロックが、保持側インスタンスのメモリー・キャッシュから要求側インスタンスへ直接転送されます。これは、ブロックをディスクに書き込むことなく、実行されます。

前述のとおり、キャッシュ・フュージョンは、ブロック・サーバー・プロセスを使用してコミットされていないトランザクションをロールバックします。その後、BSPによって、一貫読みブロックがリクエストに直接送信されます。ブロックの状態は、インスタンスが要求した時点での状態と一貫性があります。図 6-2 に、このプロセスを示します。

キャッシュ・フュージョンは、一貫読み、読み / 書き込み要求の競合に対してのみこれを実行します。これによって、ロック・ダウングレード回数およびインスタンス間通信量が大幅に削減されます。また、OLTP やハイブリッド・アプリケーションなどの、以前は Oracle Parallel Server には適していないとされていた特定のアプリケーションのスケラビリティも向上します。

図 6-2 段階的なキャッシュ・フュージョン処理



1. リクエストの FG (フォアグラウンド) プロセスは、ロック要求メッセージをマスター・ノードに送信します。要求側ノード、保持側ノードまたはまったく別のノードが、マスター・ノードとして機能できます。

2. マスター・ノードの LMD プロセスは、要求されたブロックの排他ロックを持つ保持側ノードの LMD プロセスに、ロック要求を送信します。
3. 保持側ノードの LMD プロセスは、受信メッセージを処理し、要求されたブロックの一貫読み込みコピーを準備するよう保持側インスタンスの BSP に要求します。
4. BSP は、要求されたブロックを準備し、リクエストの FG プロセスに送信します。

システム変更番号処理

システム変更番号 (SCN) は、Oracle が単一インスタンス内およびすべてのインスタンスにわたるイベントの順序付けに使用する論理タイムスタンプです。たとえば、Oracle は 1 つの SCN を各トランザクションに割り当てます。概念的には、SCN を生成するグローバル・シリアル・ポイントが存在します。ただし、実際には、SCN はパラレルで読み込みおよび生成できます。このような SCN 生成スキームの 1 つを「Lamport SCN 生成スキーム」といいます。

単一インスタンスの Oracle では、システム・グローバル領域上で、データベースを排他モードでマウントしたインスタンスから、SCN がメンテナンスおよび増加されます。Oracle Parallel Server 共有モードでは、SCN をグローバルにメンテナンスする必要があります。その実装は、プラットフォームによって異なります。SCN は、分散ロック・マネージャ、Lamport SCN スキームによって、またはハードウェア・クロックか専用 SCN サーバーを使用して処理されます。

Lamport SCN 生成の動作方法

Lamport SCN 生成スキームは、SCN をすべてのインスタンス上にパラレルで生成するため、高速かつスケラブルです。このスキームでは、ロック・メッセージを含む、インスタンス全体のすべてのメッセージによって SCN が伝達 (ピギーバック) されます。ピギーバックされた SCN によって、Oracle 内の因果関係が伝播されます。この方法で因果関係が考慮される限り、複数のインスタンスは、これらのインスタンス間で余計な通信を行わずに、SCN をパラレルで生成できます。

ほとんどのプラットフォームでは、MAX_COMMIT_PROPAGATION_DELAY がプラットフォーム固有のしきい値より大きい場合、Lamport SCN スキームが使用されます。これは、通常、デフォルトです。この値は、通常 7 秒に設定されています。インスタンス起動後にアラート・ログを検査して、Lamport SCN 生成スキームが使用されているかどうかを確認できます。この値が 7 よりも小さい場合、ロック SCN 生成スキームが使用されます。

Oracle Parallel Server の記憶域上の考慮事項

この章では、Oracle Parallel Server アプリケーションの記憶域に関する考慮事項について説明します。この章の内容は次のとおりです。

- Oracle Parallel Server 固有の記憶域に関する問題点
- 領域管理および空きリスト・グループ
- エクステンツ割当ての制御

Oracle Parallel Server 固有の記憶域に関する問題点

この項では、Oracle Parallel Server に固有の記憶域の問題点について説明します。内容は次のとおりです。

- データ・ファイル
- REDO ログ・ファイル
- ロールバック・セグメント

データ・ファイル

すべての Oracle Parallel Server インスタンスは、同じデータ・ファイルにアクセスします。データベース・ファイルの構成は、パラレル・モードおよび排他モードでは同じです。Oracle をパラレル・モードで起動するために、データ・ファイルを変更する必要はありません。

注意： Oracle Parallel Server には、『Oracle8i Parallel Server セットアップおよび構成ガイド』に記載されているとおり、データ・ファイル用に特別のネーミング規則が必要です。

パフォーマンスを改善するために、データの物理的な配置を制御できます。これによって、インスタンスは、データ・ブロックの別の集合を使用します。たとえば、空きリストによって、挿入領域を特定のインスタンスに割り当てることができます。

インスタンスは起動するたびに、すべてのオンライン・データ・ファイルへのアクセスを検証します。起動された最初の Oracle Parallel Server インスタンスは、メディアのリカバリが必要かどうかを判断できるように、すべてのオンライン・ファイルへのアクセスを検証する必要があります。その他のインスタンスは、すべてのオンライン・データ・ファイルにアクセスせずに操作できますが、検証されていないファイルを使用する試みはすべて失敗し、メッセージが生成されます。

1つのインスタンスがデータ・ファイルを追加またはオンライン化する場合、すべてのインスタンスはそのファイルへのアクセスを検証します。1つのインスタンスが、他のインスタンスがアクセスできないディスクに新しいデータ・ファイルを追加する場合、検証は失敗しますが、それらのインスタンスは実行し続けます。検証は、インスタンスが同じデータ・ファイルの別のコピーにアクセスした場合も失敗します。

どのインスタンスで検証が失敗しても、問題を診断および解決して、ALTER SYSTEM CHECK DATAFILES 文を使用してアクセスを検証してください。この文には、GLOBAL オプションがあります。これはデフォルトであり、すべてのインスタンスにオンライン・データ・ファイルへのアクセスを検証させます。この文には、カレント・インスタンスにアクセスを検証させる LOCAL オプションもあります。

ALTER SYSTEM CHECK DATAFILES は、アクセスが検証されたインスタンスがオンライン・データ・ファイルを使用できるようにします。

リカバリを実行するインスタンスが、すべての要求されたオンライン・データ・ファイルへのアクセスを検証できない限り、Oracle はインスタンス障害またはメディア障害からリカバリできません。

Oracle では、自動的に絶対ファイル番号を相対ファイル番号にマップします。Oracle Parallel Server を使用しても、これらの値には影響を及ぼしません。データ・ファイルの両方の番号を確認するには、V\$DATAFILE ビューを問い合わせてください。

REDO ログ・ファイル

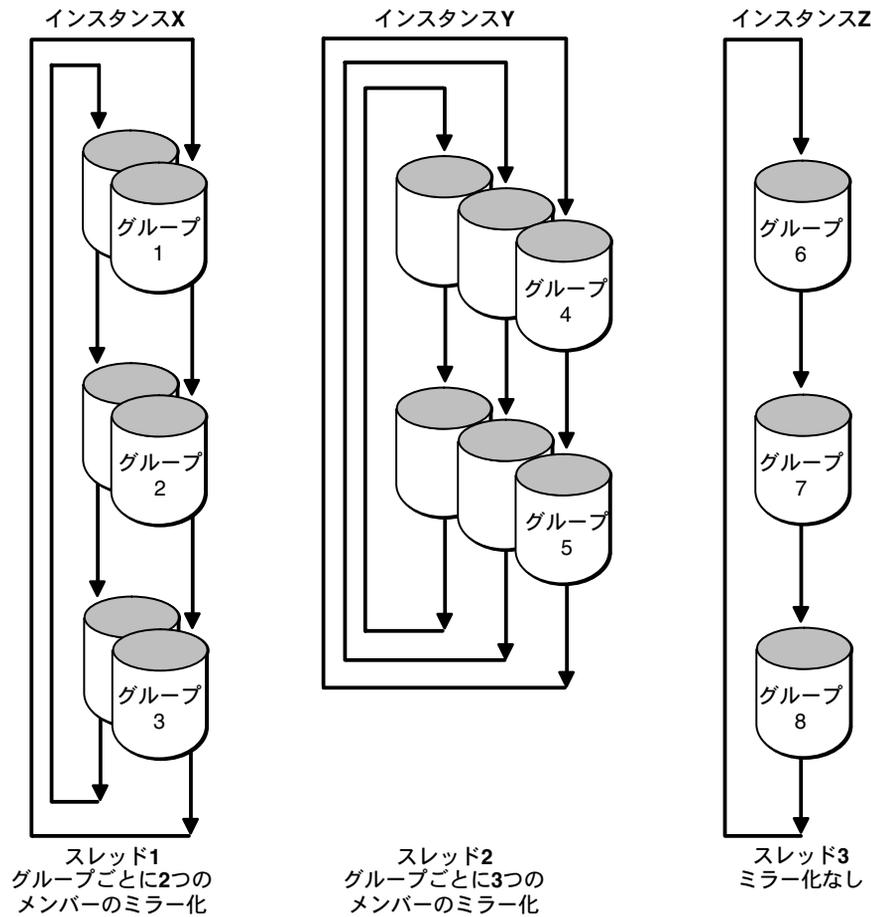
Oracle Parallel Server では、それぞれのインスタンスは独自のオンライン REDO ログ・ファイルの集合に書き込みます。単一のインスタンスが書き込んだ REDO を「REDO スレッド」といいます。それぞれのオンライン REDO ログ・ファイルは、特定のスレッド番号に対応付けられます。オンライン REDO ログがアーカイブされた場合、Oracle はスレッド番号を記録してリカバリ中に識別します。

「プライベート・スレッド」は、THREAD 句を指定した ALTER DATABASE ADD LOGFILE コマンドを使用して作成された REDO ログです。「パブリック・スレッド」は、THREAD 句を指定しない ALTER DATABASE ADD LOGFILE コマンドを使用して作成された REDO ログです。

THREAD 初期化パラメータが指定される場合、起動するインスタンスは、その値によってプライベート・スレッドとして識別されるスレッドを取得します。THREAD がデフォルトの 0 (ゼロ) である場合、そのインスタンスはパブリック・スレッドを取得します。一度取得されると、その取得インスタンスは排他的に REDO スレッドを使用します。

オンライン REDO ログ・ファイルは、多重化またはミラー化できます。多重 REDO ログは、2 つ以上のファイル・グループから構成されます。1 つのグループのすべてのメンバーは、そのグループがアーカイブされると同時に書き込まれます。図 7-1 に、Oracle Parallel Server の 3 つのインスタンスの REDO スレッドを示します。

図 7-1 REDO スレッド



- インスタンス X は、スレッド 1 を使用します。スレッド 1 には、グループ 1、2 および 3 という 3 つのオンライン REDO ログ・ファイルのグループが含まれます。スレッド 1 は、多重化されています。つまり、各グループには、REDO ログ・ファイルのコピーまたはメンバーが 2 つあります。
- インスタンス Y は、スレッド 2 を使用します。スレッド 2 には、グループ 4 および 5 という 2 つのオンライン REDO ログ・ファイルのグループが含まれます。スレッド 2 は多重化されており、各グループにはメンバーが 3 つあります。

- インスタンス Z は、スレッド 3 を使用します。スレッド 3 には、グループ 6、7 および 8 という 3 つの多重化されていないオンライン REDO ログ・ファイルのグループが含まれます。

注意： Database Configuration Assistant は、各グループに 2 つのログ・ファイル、および各インスタンスに 1 つのグループを作成します。

グループ番号は、データベース内で一意である必要があります。したがって、それらはスレッド内で一意です。ただし、グループをスレッドに、およびスレッドをインスタンスに割り当てる順序は任意です。

たとえば、[図 7-1](#) では、スレッド 1 にはグループ 1、2 および 3 が、スレッド 2 にはグループ 4 および 5 が含まれますが、かわりにグループ 2、4 および 5 をスレッド 1 に、グループ 1 および 3 をスレッド 2 に割り当てることもできます。V\$LOGFILE ビューは、各 REDO ログ・ファイルに対応付けられているグループ番号を表示します。

スレッドごとに異なるグループ数およびメンバー数を持つことは可能ですが、管理を容易にするため、すべてのスレッドを共通の標準に基づいて構成することをお勧めします。

Oracle Parallel Server の異なるインスタンスは、グループごとに異なるミラー化度またはメンバー数を持つことができます。異なるインスタンスは、異なる数のグループを持つこともできます。たとえば、1 番目のインスタンスはそれぞれ 2 つのメンバーを持つ 3 つのグループを、2 番目のインスタンスは 4 つの多重化されていないログ・ファイルを、3 番目のインスタンスはそれぞれ 4 つのメンバーを持つ 2 つのグループを持つことが可能です。このような構成は管理が不便ですが、システムの潜在能力を最大限に活用するためには必要な場合もあります。

各インスタンスは、少なくとも 2 つのグループのオンライン REDO ログ・ファイルを持つ必要があります。カレント・グループが一杯になると、インスタンスが次のログ・ファイル・グループへの書込みを開始します。ログ・スイッチで情報が制御ファイルに書き込まれます。この制御ファイルを使用して、一杯になったグループおよびそのスレッド番号をアーカイブされた後に識別できます。

制御ファイルで関連情報を保持できる REDO ログ・ファイルの数は、CREATE DATABASE 文の MAXLOGHISTORY オプションの値によって制限されます。グループごとに 1 つのメンバーのみ必要です。Oracle Parallel Server では、MAXLOGHISTORY の値を通常単一インスタンスの Oracle で設定するよりも高く設定してください。これは、Oracle Parallel Server では複数の REDO ログ・ファイルの履歴を追跡する必要がありますためです。

注意： MAXLOGHISTORY は、可用性要件が非常に高いサイトで役立ちます。このオプションは、特に多数のインスタンスおよびログ・ファイルがある場合、リカバリを管理する上で有効です。

参照： 多重 REDO ログ・ファイルの詳細は、『Oracle8i 概要』を参照してください。

ロールバック・セグメント

この項では、Oracle Parallel Server に関連するロールバック・セグメントについて説明します。

- Oracle Parallel Server のロールバック・セグメント
- ロールバック・セグメントを制御するパラメータ
- パブリックおよびプライベート・ロールバック・セグメント
- インスタンスによるロールバック・セグメントの取得方法

Oracle Parallel Server のロールバック・セグメント

ロールバック・セグメントには、Oracle が読取り一貫性を維持するため、およびロールバックまたは異常終了したトランザクションが行った変更の取消しを可能にするために必要な情報が含まれます。Oracle Parallel Server 内の各インスタンスは、SYSTEM ロールバック・セグメントを共有して使用し、インスタンスごとに少なくとも1つの専用ロールバック・セグメントを必要とします。

プライベート・ロールバック・セグメントおよびパブリック・ロールバック・セグメントは、オフラインにされるまで、または ROLLBACK_SEGMENTS パラメータへの指定によりそれを取得したインスタンスが停止されるまで、両方ともインスタンス起動時に取得でき、取得インスタンスによって排他的に使用できます。プライベート・ロールバック・セグメントは、特定のインスタンスに対して一意です。他のインスタンスはそれらを使用できません。パブリック・ロールバック・セグメントは、追加のロールバック・セグメントが必要なインスタンスが起動してそれを取得およびオンラインにするまで、オフラインであり、どのインスタンスにも使用されません。一度オンラインにすると、それを取得したインスタンスは排他的にパブリック・ロールバック・セグメントを使用します。

1つの指定されたロールバック・セグメントに書き込めるインスタンスは1つのみです (SYSTEM ロールバック・セグメントを除きます)。ただし、他のインスタンスは、そのセグメントから読み込んで、読取り一貫性スナップショットを作成したりインスタンス・リカバリを実行したりできます。

Oracle Parallel Server には、少なくとも同時に実行されるインスタンスの最大数に1を加えた数のロールバック・セグメントが必要です。この加えられた1は SYSTEM ロールバック・セグメントになります。インスタンスは、パブリックかプライベートかにかかわらず、少なくとも1つのロールバック・セグメントに排他的にアクセスできなければ、共有モードで起動できません。

どの表領域にも新しいロールバック・セグメントを作成できます。ロールバック・データおよび表データ間の競合を削減するには、表データと別の表領域にロールバック・セグメントを作成します。これによって表領域をオフラインにするのも容易になります。表領域は、アクティブ・ロールバック・セグメントを含むとオフラインにされないためです。

一般に、記憶域パラメータ INITIAL および NEXT に同じ値を指定することによって、すべてのロールバック・セグメントのエクステントを同じサイズにします。

データ・ディクショナリ・ビュー DBA_ROLLBACK_SEGS は、各ロールバック・セグメントの名前、セグメント ID 番号および所有者 (PUBLIC またはその他) を示します。

参照： ロールバック・セグメントの競合、およびロールバック・セグメントの追加によるパフォーマンスへの影響については、『Oracle8i 管理者ガイド』を参照してください。

ロールバック・セグメントを制御するパラメータ

次の初期化パラメータは、ロールバック・セグメントの使用を制御します。

ROLLBACK_SEGMENTS	インスタンスが起動時に取得するロールバック・セグメントの名前を指定します。
GC_ROLLBACK_LOCKS	ロールバック・エントリを含むブロックに対する競合を削減するための、インスタンス・ロックを確保します。特に、遅延ロールバック・セグメントのインスタンス・ロックを確保します。遅延ロールバック・セグメントには、オフラインにされた表領域におけるトランザクションのロールバック・エントリが含まれます。

参照： データ・ブロック、エクステントおよびセグメントの詳細は、『Oracle8i 概要』を参照してください。

パブリックおよびプライベート・ロールバック・セグメント

パブリックおよびプライベート・ロールバック・セグメントには、パフォーマンスの違いはありません。ただし、プライベート・ロールバック・セグメントは、インスタンスとロールバック・セグメントの一致をより詳細に制御します。これによって、異なるインスタンスのロールバック・セグメントを異なるディスク上に格納して、パフォーマンスを改善できます。したがって、高パフォーマンスのシステムでディスク競合を削減するには、プライベート・ロールバック・セグメントを使用してください。

パブリック・ロールバック・セグメントは、追加のロールバック・セグメントを必要とする、どのインスタンスによっても取得できるロールバック・セグメントのプールを形成します。ただし、インスタンスが停止されると同時に起動されたときは、パブリック・ロールバック・セグメントを使用すると不利な場合もあります。たとえば、インスタンス X は、停止してパブリック・ロールバック・セグメントを解放します。インスタンス Y は、起動して解放されたロールバック・セグメントを取得します。最終的に、インスタンス X は、起動しても元のロールバック・セグメントを取得できません。パブリック・ロールバック・セグメントは、TRANSACTIONS および TRANSACTIONS_PER_ROLLBACK_SEGMENTS が適切に設定されていない場合にも起動時に取得されます。

パブリック・ロールバック・セグメントを使用して、領域使用率を改善できます。月ごとに異なるインスタンス上で実行する長時間実行トランザクション用に、1つの大きなパブリック・ロールバック・セグメントのみを作成した場合、そのロールバック・セグメントはオフラインにされて別のインスタンスでオンラインにされる、すなわちインスタンス間を移動して、より作業負荷の大きいインスタンスで使用されます。

デフォルトでは、ロールバック・セグメントはプライベートであり、このセグメントをパラメータ・ファイル内に指定するインスタンスによって使用されます。プライベート・ロールバック・セグメントを指定するには、ROLLBACK_SEGMENTS パラメータを使用します。

一度パブリック・ロールバック・セグメントがインスタンスによって取得されると、そのインスタンスによって排他的に使用されます。

一度作成されると、パブリックおよびプライベート・ロールバック・セグメントのどちらも、ALTER ROLLBACK SEGMENT コマンドを使用してオンラインにできます。

注意： インスタンスには、少なくとも1つのロールバック・セグメントが必要です。ロールバック・セグメントがない場合、インスタンスは起動できません。

インスタンスによるロールバック・セグメントの取得方法

インスタンスが起動すると、TRANSACTIONS および TRANSACTIONS_PER_ROLLBACK_SEGMENT 初期化パラメータを使用して、次の等式に従って、取得するロールバック・セグメントの数を判断します。

$$\frac{\text{TRANSACTIONS}}{\text{TRANSACTIONS_PER_ROLLBACK_SEGMENT}} = \text{total_rollback_segments_required}$$

total_rollback_segments_required の値は、切り上げられます。

起動時に、インスタンスは、次のステップを実行してロールバック・セグメントの取得を試みます。

- まず、インスタンスは、ROLLBACK_SEGMENTS 初期化パラメータによって指定されたすべてのプライベート・ロールバック・セグメントを取得します。
total_private_rollback_segments の数値が *total_rollback_segments_required* より大きい場合、ロールバック・セグメント取得のためのそれ以上の処理は行われません。
- 初期化ファイルがプライベート・ロールバック・セグメントを指定しない場合、そのインスタンスは、パブリック・ロールバック・セグメントの取得を試みます。
- *total_private_rollback_segments* が *total_rollback_segments_required* より小さい場合、インスタンスはパブリック・ロールバック・セグメントを取得してその差の補完を試みます。

- 1つのプライベート・ロールバック・セグメントのみが指定および取得される場合、または1つのパブリック・ロールバック・セグメントが取得される場合、その1つのロールバック・セグメントでは *total_rollback_segments_required* に満たない場合でも、そのインスタンスは起動します。この場合、Oracle はメッセージを生成します。
- プライベート・ロールバック・セグメントをインスタンス起動時にオンラインにできない場合、起動は失敗し、Oracle はメッセージを生成します。

領域管理および空きリスト・グループ

この項では、領域管理および空きリスト・グループの概念を説明します。内容は次のとおりです。

- Oracle による空き領域の処理方法
- 空きリストおよび空きリスト・グループ
- 空きリスト・グループを使用したデータのパーティション化
- インスタンス、ユーザーおよびロックと空きリスト・グループの対応付け
- 空き領域管理のための SQL オプション

Oracle による空き領域の処理方法

Oracle Parallel Server では、個別のインスタンス上で実行するトランザクションが、同じ表内のデータを同時に挿入および更新できます。これは、新しいレコード用の空き領域を検索するための競合を伴うことなく実行されます。ただし、この機能を利用するには、この項で説明するいくつかの構造を使用して、データベースの空き領域を正しく管理する必要があります。

Oracle では、行が元のブロック内の使用可能領域を超える場合があるトランザクションのために、使用可能領域のあるブロックを追跡します。Oracle は、これを表、クラスタ、索引などのデータベース・オブジェクトごとに実行します。空き領域を必要とするトランザクションは、ブロックの空きリストを調査できます。空きリストに十分な格納領域のあるブロックがない場合、Oracle は新しいエクステントを割り当てます。

Oracle が表に新しいエクステントを割り当てるときは、エクステントのブロックがマスター空きリストに追加されます。ただし、新しく割り当てられたエクステントに含まれる空き領域は、どの空きリスト・グループにも再割当てできないため、最終的には Oracle Parallel Server 上の複数のインスタンス間で空き領域に対する競合が発生します。エクステントをインスタンスに明確に割り当てると、空き領域をより強力に制御できます。この方法で、空き領域に対する競合を削減できます。

セグメント、エクステントおよび最高水位標

セグメントは、論理データベース記憶域の単位です。Oracle では、エクステントという、より小さい単位にセグメントの領域が割り当てられます。エクステントは、特定のタイプの情報を格納するために割り当てられた連続するデータベース・ブロックの特定の番号です。したがって、セグメントは、特定のタイプのデータ構造体のために割り当てられたエクステントの集合で構成されます。たとえば、各表のデータはそのデータのデータ・セグメントに格納されますが、各索引のデータはそのデータの索引セグメントに格納されます。

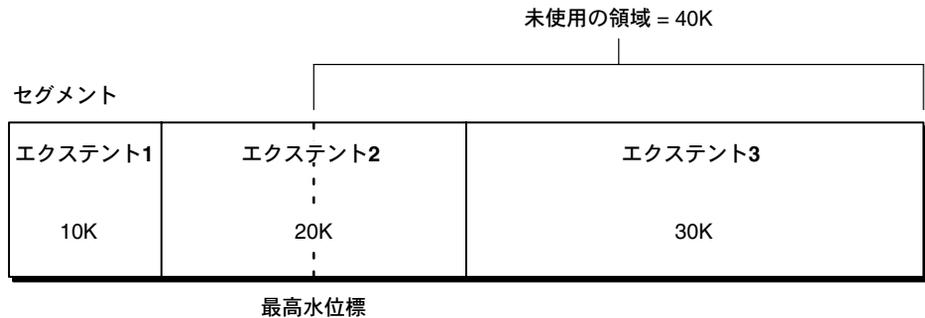
セグメントのエクステントは、同じ表領域に格納されます。ただし、これらはディスク上で連続しているとは限りません。セグメントはファイルにまたがることができますが、個々のエクステントはできません。

追加のエクステントを割り当てることはできますが、ブロック自体は別々に割り当てられません。エクステントを特定のインスタンスに割り当てる場合、ブロックは直ちに空きリストに割り当てられます。ただし、エクステントが特定のインスタンスに割り当てられないときは、ブロック自体は、最高水位標が移動する場合にのみ割り当てられます。

最高水位標は、セグメント内の使用済領域と未使用領域間の境界です。既存の空きリストでは満たすことのできない新しい空きブロックへの要求が受信されると、最高水位標が指すブロックは使用済ブロックとなり、最高水位標は次のブロックに移動します。つまり、最高水位標の左側のセグメント領域は使用済であり、最高水位標の右側の領域は未使用です。

図 7-2 に、それぞれ 10K、20K および 30K の領域を含む 3 つのエクステントで構成されるセグメントを示します。最高水位標は、2 番目のエクステントの中間にあります。したがって、セグメントは最高水位標の左に 20K の使用済領域を、右に 40K の未使用領域を含みます。

図 7-2 最高水位標



参照： セグメントおよびエクステントの詳細は、『Oracle8i 概要』を参照してください。

空きリストおよび空きリスト・グループ

単一インスタンスの Oracle では、ブロックに対する競合を削減するために、複数の空きリストが使用されます。各表領域には、空き領域のあるデータ・ブロックを識別する空きリストがあります。Oracle では、表やクラスタなどのデータベース・オブジェクトへの挿入および更新が行われた場合、空き領域のあるブロックが使用されます。

空きリストにあるブロックは、PCTFREE より大きい空き領域を含みます。PCTFREE は、既存の行の更新用に確保されたブロックが占める割合です。一般に、データベース・オブジェクト用のプロセス空きリストに含まれるブロックは、PCTFREE および PCTUSED 制約を満たす必要があります。

表、索引およびクラスタを作成する場合、FREELISTS パラメータを設定することによって、空きリスト数を指定できます。FREELISTS パラメータの最大値は、システム上の Oracle ブロック・サイズによって異なります。さらに、オーバーヘッドを削減するには、各空きリストの1つのブロックに一定のバイト数を格納する必要があります。

空きリスト・グループ内には、次に示す2つの空きリストのサブセットがあります。

- マスター空きリスト。表のすべてのエクステントからの使用可能な領域を含むブロックのリストです。
- トランザクション空きリスト。コミットされていないトランザクションによって解放されたブロックのリストです。

Parallel Server には複数のインスタンスがあるため、空きリストのみでは競合問題を解決できません。ただし、空きリスト・グループによって、インスタンス間の ping を効率的に削減できます。

注意： 確保された領域および空きリストごとに必要なバイト数は、プラットフォームによって異なります。詳細は、システム固有の Oracle マニュアルを参照してください。

空きリスト・グループ

空きリスト・グループは、1つ以上のインスタンスが使用する空きリストの集合です。各空きリスト・グループは、表およびクラスタへの挿入または更新時に空きデータ・ブロックを提供し、起動時にインスタンスと対応付けられます。デフォルトでは、1つの空きリスト・グループのみ使用可能です。これは、1つのオブジェクトのすべての空きリストが、セグメント・ヘッダー・ブロックに常駐することを意味します。空きリスト・グループは、すべてのデータベース・オブジェクトでサポートされます。

複数の空きリストが Oracle Parallel Server 環境で単一ブロックに常駐する場合、空きリストがあるブロックには、ping、またはすべてのインスタンス間の強制読み込み / 書き込みが行われる場合があります。これを回避するには、空きリストを個別のグループにグループ化し、各グループを1つのインスタンスに割り当てます。これによって、各インスタンスは、空きリストを含む独自のブロックを持ちます。各インスタンスは独自の空きリストを使用するため、空きリストを含む同一ブロックにアクセスするためのインスタンス間の競合はありません。

インスタンス間の競合は、異なるインスタンスのトランザクションがデータを同じ表に挿入する場合に発生します。これは、空きリスト・グループを定義しない場合、すべての空きリストがセグメント・ヘッダーに保持されるためです。空きリストは、ブロックの共有プールからのものである場合があります。そうでない場合は、複数の空きリストをパーティション化し、ファイル内の特定のエクステン트를オブジェクトに割り当てることができます。

注意： Oracle Parallel Server では、常に空きリスト・グループおよび空きリストを使用してください。

セグメント・ヘッダーおよび空きリストに対する競合回避

同時性の高い環境では、空きリストを含むセグメント・ヘッダーに対する競合が発生する可能性があります。

- 空きリスト・グループが存在する場合、セグメント・ヘッダーは共有の空きリストのみを指します。さらに、どの空きリスト・グループのブロックも、独自の空きリストを指すポインタを含みます。
- 空きリスト・グループが存在しない場合、セグメント・ヘッダーが空きリストを指すポインタを含みます。

複数インスタンス環境では、[図 7-3](#) に示すように、空きリストが、空きデータ・ブロックを使用可能なエクステントから別のインスタンスに提供します。複数の空きリストをパーティション化して、そのエクステントを特定のデータベース・インスタンスに割り当てることができます。各インスタンスは1つ以上の空きリスト・グループにハッシュされ、各グループのヘッダー・ブロックは空きリストを指します。空きリスト・グループがない場合、すべてのインスタンスは、空きリストにアクセスするためにセグメント・ヘッダー・ブロックを読み込む必要があります。

図 7-3 セグメント・ヘッダーに対する競合

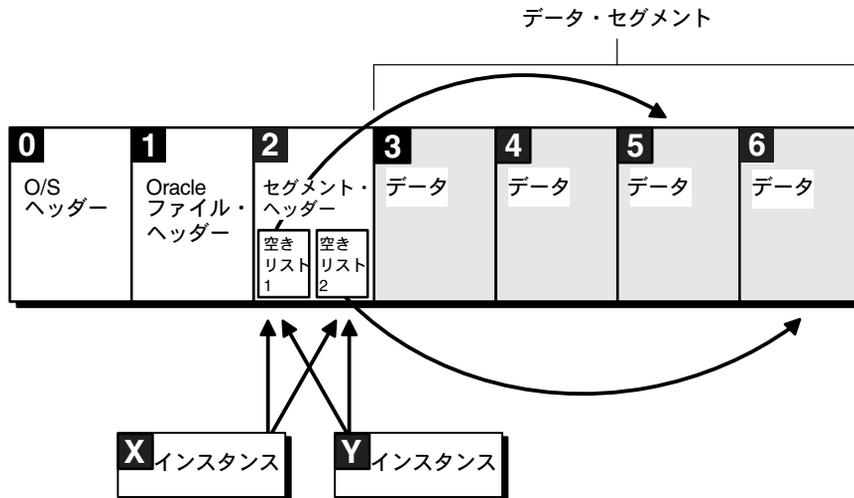
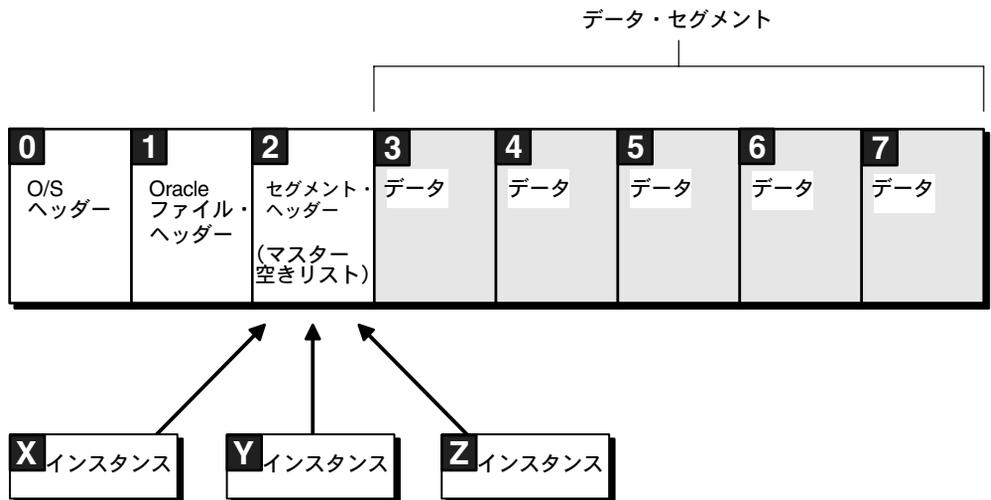


図 7-4 に、マスター空きリストがセグメント・ヘッダー・ブロックに格納されるファイルのブロックを示します。3つのインスタンスが、このブロックを強制読み込みして空き領域を取得します。1つの空きリストしかないため、挿入点は1つしかありません。空きリスト・グループは、この挿入点を複数のブロックに分散して競合を削減します。各ブロックへのアクセス頻度は減少します。

図 7-4 マスター空きリストに対する競合



ローカル管理表領域

ローカル管理表領域も、ディクショナリ競合を回避するために有用です。これは、表領域内のセグメントのソート、表の作成や削除などのために Oracle Parallel Server が多くの領域管理を実行する場合に実行されます。ローカルで管理された表領域はこうした競合を排除し、この場合の Oracle Parallel Server での効果は、単一インスタンスの Oracle での効果をはるかに上回ります。

空きリスト・グループの例

この項では、空きリスト・グループの2つの例を説明します。

- 基本的な空きリスト・グループの例
- 複雑な空きリスト・グループの例

基本的な空きリスト・グループの例

図 7-5 は、1つの表の空き領域を、1つのマスター空きリストおよび2つの空きリスト・グループにパーティション化することを示しています。2つの空きリスト・グループは、それぞれ3つの空きリストを含みます。この例には、その中で削除が発生する、適切にパーティション化されたアプリケーションが含まれています。図に示されているマスター空きリストは、この特定の空きリスト・グループ用のマスター空きリストです。

この表は、1つの初期エクステントで作成された後に、エクステント2および5がインスタンス X に、エクステント3および4がインスタンス Y に、およびエクステント6が自動的に不特定のインスタンスに割り当てられました。次のことに注意してください。

- エクステント1および6にある初期割当ての濃い灰色のブロックは、空きブロックのマスター空きリストを表します。
- エクステント2および5にある薄灰色のブロックは、空きリスト・グループ X 内の使用可能な空き領域を表します。
- エクステント3および4にある中灰色のブロックは、空きリスト・グループ Y 内の使用可能な空き領域を表します。
- エクステント5は新たに割り当てられるため、そのすべてのブロックは空きリスト・グループ X 内にあります。
- 黒いブロックは、削除によって解放されたブロックを表します。これらのブロックは、空きリスト・グループ X および Y に戻されます。
- 白いブロックは、挿入するのに十分な空き領域を含みません。

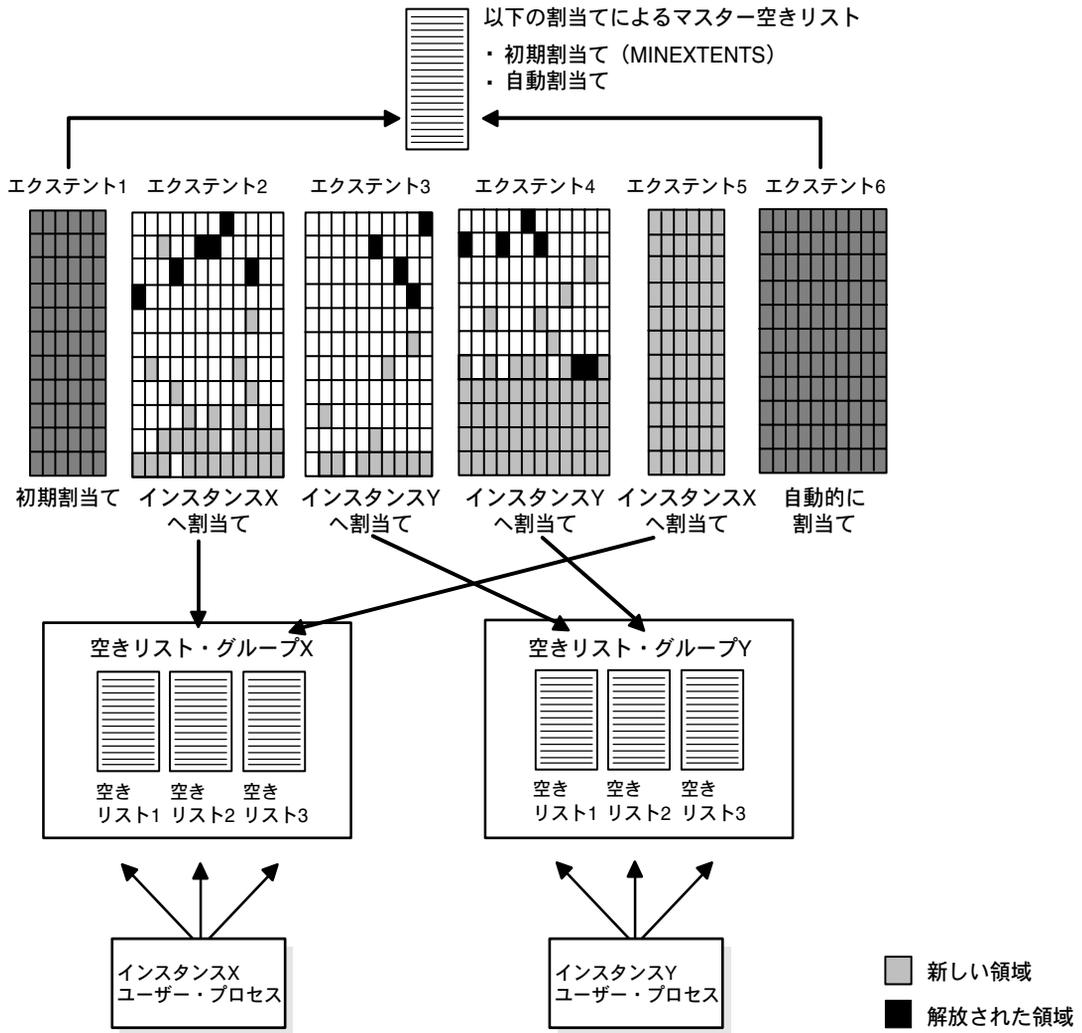
インスタンス X 上で実行する各ユーザー・プロセスは、グループ X にある空きリストの1つを使用し、インスタンス Y 上の各ユーザー・プロセスは、グループ Y にある空きリストの1つを使用します。さらに多くのインスタンスが起動する場合、そのユーザー・プロセスは、インスタンス X または Y と空きリストを共有します。

複雑な空きリスト・グループの例

図 7-5 では単純なケースが、エクステン트가永続的にインスタンスに割り当てられるわけではなく、また 1 つのインスタンスに割り当てられる領域は別のインスタンスが使用できないことを考慮すると、より複雑になります。各空きリスト・グループには、独自のマスター空きリストがあります。割当て後、ブロックの中には、そのグループのマスター空きリストまたはプロセス空きリストに移動するものもありますが、まったく空きリストに属さないものもあります。アプリケーションが完全にパーティション化される場合、一度ブロックが特定のインスタンスに割り当てられると、ブロックはそのインスタンスに割り当てられたままになります。ただし、アプリケーションが完全にパーティション化されない場合、ブロックはインスタンス間を移動できます。

インスタンス Y のブロックが一杯になって、空きリストからそのブロックを取り除いた後に、インスタンス X がそのブロックを解放する場合を考えてみます。そのブロックは、ブロックを解放したインスタンス X の空きリストに移動します。インスタンス Y が領域を必要としても、このブロックを再利用することはできません。インスタンス Y は、自身の空きリスト・グループからしか空きリストを取得できません。

図 7-5 表の空きリスト・グループ



空きリスト・グループを使用したデータのパーティション化

一般に、すべての表は同じ数の空きリスト・グループを持つ必要があります。ただし、1つのグループ内の空きリスト数は、各表のアクティビティのタイプおよび量によって異なる場合があります。

空き領域のパーティション化によって、特に、同時挿入または複数のインスタンスから新しい領域を要求する更新が多量に実行されるアプリケーションのパフォーマンスを改善できます。もちろん、パフォーマンスの改善は、オペレーティング・システム、ハードウェア、データのブロック・サイズなどにも依存します。

複数インスタンス環境では、複数の空きリストおよび空きリスト・グループに関する情報は、インポート時には保存されません。エクスポートおよびインポートを使用してデータをバックアップおよびリストアする場合は、再度パーティション化できるようにデータをインポートすることは困難です。

Oracle による空きリスト・グループのパーティション化方法

実際の空きリスト・グループのブロックは、空きリスト・グループ数による Oracle プロセス ID のハッシュによって判断されます。たとえば、3つのインスタンスおよび35個の空きリスト・グループがある場合、インスタンス1が最初の12個の空きリスト・グループを処理し、インスタンス2が次の12個、およびインスタンス3が残る11個の空きリスト・グループを処理します。

インスタンス、ユーザーおよびロックと空きリスト・グループの対応付け

この項の内容は次のとおりです。

- インスタンスと空きリストの対応付け
- ユーザー・プロセスと空きリストの対応付け
- PCM ロックと空きリストの対応付け

インスタンスと空きリストの対応付け

データのパーティション化によって、データ・ブロックに対する競合を削減できます。1つの空きリスト内の複数のグループを扱うことが多い PCM ロックは、その空きリスト・グループを使用するインスタンス専用に保持される傾向があります。これは、データを変更するインスタンスが、通常、他のインスタンスよりもそのデータを再使用する可能性が高いためです。ただし、複数のインスタンスが同じエクステントから空き領域を取る場合、それらのインスタンスは、挿入したデータを変更するときに、そのエクステントにあるブロックに対して競合する可能性が高まります。

既存空きリスト・グループへの新しいインスタンスの割当て

MAXINSTANCES が表またはクラスタ内の空きリスト・グループ数より大きい場合、1つのインスタンスの番号は、次に対応付けられる空きリスト・グループにマップされます。

instance_number modulo number_of_free_list_groups

「modulo」(または剰余を表す「rem」)は、剰余値を計算することによって、どの空きリスト・グループを使用する必要があるかを判断する計算式です。次の例では、2つの空きリスト・グループおよび10個のインスタンスがあります。インスタンス6がどの空きリスト・グループを使用するかを判断する計算式は、 $6 \bmod 2 = 0$ となります。6を2で割ると3で剰余0のため、インスタンス6は空きリスト・グループ0を使用します。同様に、インスタンス5は、 $5 \bmod 2 = 1$ であるため、空きリスト・グループ1を使用します。5を2で割ると剰余1となるためです。

MAXINSTANCES より大きい空きリスト・グループがある場合、異なるハッシュ・メカニズムが使用されます。複数のインスタンスが1つの空きリスト・グループを共有する場合、その空きリスト・グループを共有するインスタンスに明確に割り当てられたすべてのエクステントへのアクセスを共有します。

FREELIST GROUPS および MAXINSTANCES

比較的ノード数の少ないシステムでは、表の FREELIST GROUPS オプションは、一般に CREATE DATABASE の MAXINSTANCES オプションと同じ値を持つ必要があります。これによって、データベースに同時にアクセスできるインスタンス数を制限します。ただし、超並列 (MPP) システムでは、MAXINSTANCES は FREELIST GROUPS より数倍も大きくなる可能性があるため、多数のインスタンスが 1 つの空きリスト・グループを共有する場合があります。

参照： インスタンス、ユーザーおよびロックと空きリスト・グループの対応付けに関する詳細は、『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を参照してください。

ユーザー・プロセスと空きリストの対応付け

ユーザー・プロセスは、Oracle プロセス ID に基づいてプロセス空きリストと対応付けられます。各ユーザー・プロセスは、それが実行しているインスタンスの空きリスト・グループにある 1 つの空きリストへのみアクセスできます。各ユーザー・プロセスは、マスター空きリストの空きブロックにアクセスできます。

表に複数の空きリストはあるが、複数の空きリスト・グループはないか、インスタンス数よりも少ない空きリスト・グループしかない場合、各空きリストは、他のインスタンスからのユーザー・プロセスによって共有されます。

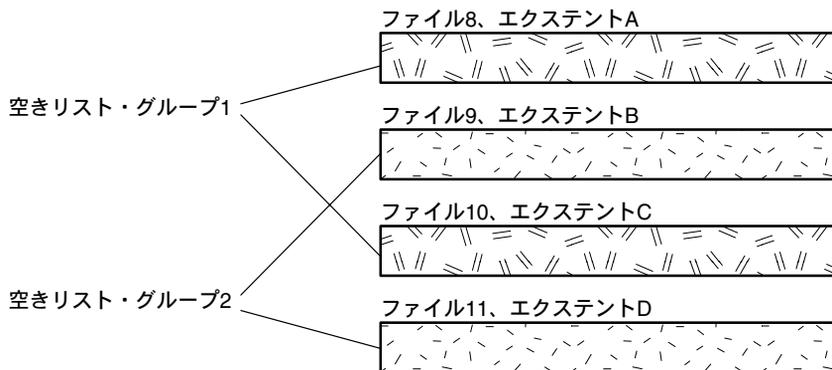
PCM ロックと空きリストの対応付け

表の各エクステントが個別のデータ・ファイルにある場合、GC_FILES_TO_LOCKS パラメータを使用して特定の PCM ロック範囲を各エクステントに割り当てることができます。これによって、PCM ロックの各集合を、1 つの空きリスト・グループのみと対応付けることができます。

図 7-6 に、別々のファイル内の複数エクステントを示します。GC_FILES_TO_LOCKS パラメータは、10 個のロックをファイル 8 および 10 に、また 10 個のロックをファイル 9 および 11 に割り当てます。エクステント A および C は同じ空きリスト・グループに、またエクステント B および D は別の空きリスト・グループにあります。1 つの PCM ロック集合はファイル 8 および 10 に、また別の PCM ロック集合はファイル 9 および 11 に対応付けられています。ファイル 8 および 10、またはファイル 9 および 11 のように、同じ空きリスト・グループ内にあるファイルには別々のロックは必要ありません。

図 7-6 エクステントおよび空きリスト・グループ

GC_FILES_TO_LOCKS = 8, 10:10; 9, 11:10

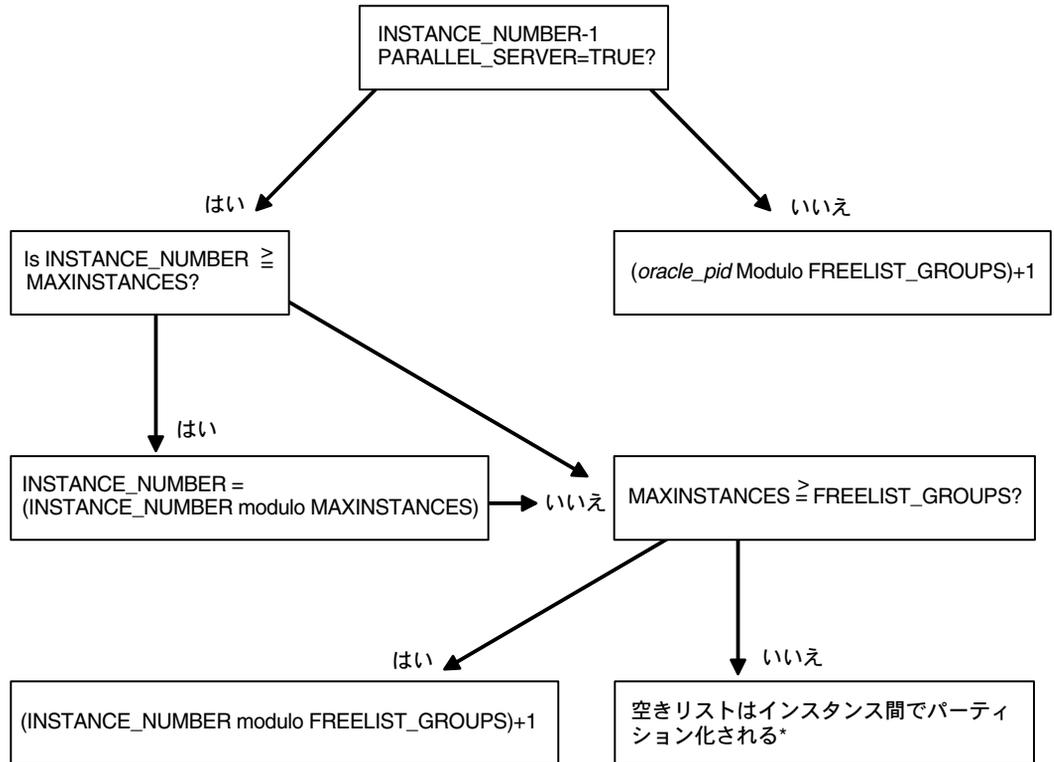


この例では、読み込みおよび書き込みの両方が完全にパーティション化されていると仮定します。複数のインスタンスがブロックを更新する場合、ファイルごとに複数のロックを持つことによって、強制読み込みおよび書き込みを最小化することが望ましいです。これは、共有ロックを使用しても、ロックされたブロックを1つでも別のインスタンスが読み込もうとすると、ロックが保持するすべてのブロックが強制読み込みされるためです。

Oracle によるインスタンスへの空きリストおよび空きリスト・グループの割当て方法

図 7-7 に、空きリストおよび空きリスト・グループがインスタンスに割り当てられる方法を示します。

図 7-7 空きリストおよび空きリスト・グループの割当て方法



ALTER SESSION INSTANCE_NUMBER 文を使用して、インスタンス番号値を MAXINSTANCES 値より大きくすることができます。図 7-7 は、これをどのように考慮するかを示しています。空きリスト・グループの割当てのための内部計算を実行するため、インスタンス番号は MAXINSTANCES 値の範囲内に戻されます。

空き領域管理のための SQL オプション

いくつかの SQL オプションによって、表、クラスタおよび索引用に空きリストおよび空きリスト・グループを割り当てることができます。オブジェクト用の新しい領域を特定のデータ・ファイルから取ることを明示的に指定できます。空き領域を特定の空きリスト・グループに対応付け、次にその空きリスト・グループを特定のインスタンスに対応付けることもできます。

こうした SQL 文には、次のものが含まれます。

```
CREATE [TABLE | CLUSTER | INDEX]
    STORAGE
    FREELISTS
    FREELIST GROUPS
ALTER [TABLE | CLUSTER | INDEX]
    ALLOCATE EXTENT
    SIZE
    DATAFILE
    INSTANCE
```

これらの SQL オプションを初期化パラメータ `INSTANCE_NUMBER` とともに使用して、データ・ブロックをインスタンスに対応付けることができます。

参照： これらの文の完全な構文については、『Oracle8i SQL リファレンス』を参照してください。

エクステント割当ての制御

この項の内容は次のとおりです。

- [新しいエクステントの自動割当て](#)
- [新しいエクステントの事前割当て](#)
- [ロック境界でのブロックの動的割当て](#)

行が表に挿入され、新しいエクステントの割当てが必要な場合、GC_FILES_TO_LOCKS パラメータの *!blocks* によって指定されるように、一定の数の連続ブロックが、インスタンスに対応付けられた空きリスト・グループに割り当てられます。表またはクラスタが最初に作成された時に割り当てられるエクステント、および自動的に割り当てられる新しいエクステントは、それらのブロックをマスター空きリストまたは最高水位標より上の領域に追加します。

新しいエクステントの自動割当て

インスタンスを指定しないでエクステントを明示的に割り当てる場合、またはシステムが領域不足のため（最高水位標をそれ以上進められない）エクステントが自動的にセグメントに割り当てられる場合、その新しいエクステントは未使用領域の一部になります。このエクステントは、エクステント・マップの最後に置かれます。これは、最高水位標が現在新しいエクステントの左のエクステントにあることを意味します。したがって、新しいエクステントは、最高水位標の上に追加されます。

新しいエクステントの事前割当て

新しいエクステントの割当てを制御するための2つのオプションがあります。

- [空きリスト・グループへのエクステントの事前割当て](#)
- [ロック境界でのブロックの動的割当て](#)

空きリスト・グループへのエクステントの事前割当て

エクステントの事前割当ては、Oracle によるエクステントの自動割当てを防ぐという問題への静的アプローチです。空きリスト・グループを持つ表にエクステントを事前に割り当てることができます。これは、すべての空きブロックが空きリスト内にフォーマットされたことを意味します。これらの空きリストは、エクステントを事前に割り当てているインスタンスの空きリスト・グループ内に常駐します。挿入でのすべての ping を大幅に削減するためデータのパーティション化が必要な場合、またはサイズの増大が見込まれるオブジェクトに対処する必要がある場合、このアプローチは有効です。

注意： false ping を完全に排除することはできません。

ロック境界でのブロックの動的割当て

拡張に対応するためには、ブロックを空きリスト・グループに動的に割り当てる方法の方がエクステントの事前割当てよりも効果的です。GC_FILES_TO_LOCKS の *!blocks* オプションを使用して、ロック境界内の最高水位標から空きリストにブロックを動的に割り当てることができます。この方法では、セグメント・ヘッダーへのすべての ping が排除されるわけではありません。かわりに、この方法では必要に応じてブロックが割り当てられるため、エクステントを事前に割り当てる必要がありません。

ロックはインスタンスが所有するため、ブロックはインスタンスごとに割り当てられます。これが、ブロックが空きリスト・グループに割り当てられる理由です。インスタンス内では、ブロックを他の空きリストに割り当てることができます。

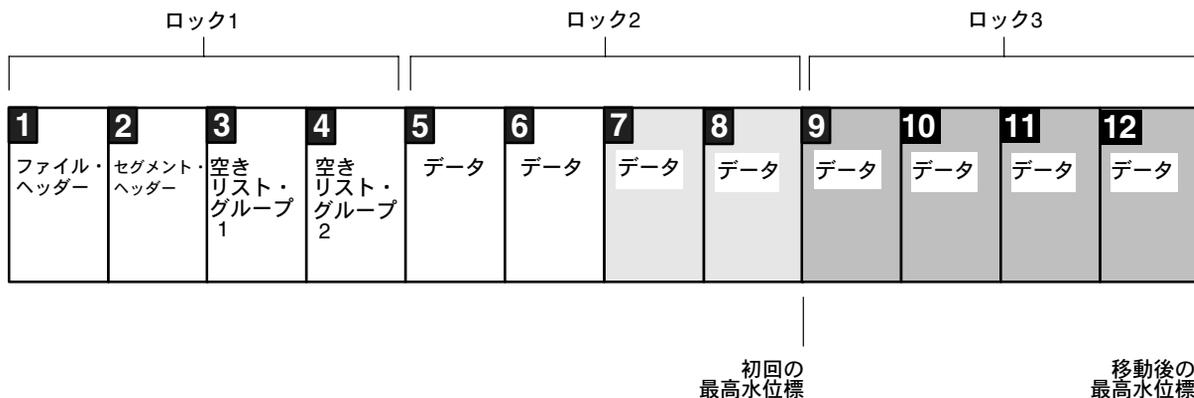
この方法を使用すれば、*!blocks* 値を明示的に割り当てるか、または依然として既存 PCM ロックによって処理されている新しいブロックの均衡を保つことができます。後者を選択する場合、依然として、他のインスタンスへの割当てによって既存 PCM ロックに対する競合が発生する場合がありますことに注意してください。PCM ロックが複数のブロック・グループを処理する場合、依然として、そのロックによって処理されるすべてのブロックを不必要に強制読み込みおよび書き込みする場合があります。

セグメントの最高水位標の移動

セグメントの最高水位標は、セグメント内で割り当てられたブロック数の現在の限界です。エクステントを動的に割り当てている場合、最高水位標はロック境界でもあります。ロック境界およびエクステント内で一度に割り当てられるブロック数は、一致する必要があります。この値は、すべてのインスタンスについて同じである必要があります。

ロックごとに4つのブロック (14) がある次の例について考えてみます。ロックは、そのブロック内容が入力される前に割り当てられています。ロック 2 によって保持されたデータ・ブロック D2 が一杯となり、4つのブロックの別の範囲が割り当てられた場合、そのロック境界内で適合するブロック数のみが割り当てられます。この場合、これにはブロック 7 および 8 が含まれます。これらは両方ともカレント・モードのロックによって保護されています。最高水位標が 8 にあり、インスタンス 2 がブロック範囲を割り当てるときは、ロック 3 によって処理される 9 ~ 12 のすべてのブロックが割り当てられます。インスタンス 1 が次にブロックを割り当てるとき、ロック 4 によって処理されるブロック 13 ~ 16 を取ります。

図 7-8 ブロックの割当てに伴って移動する最高水位標のあるファイル



例 この例では、GC_FILES_TO_LOCKS が両方のインスタンスに対して次のように設定されていると仮定します。

```
GC_FILES_TO_LOCKS = "1000!5"
```

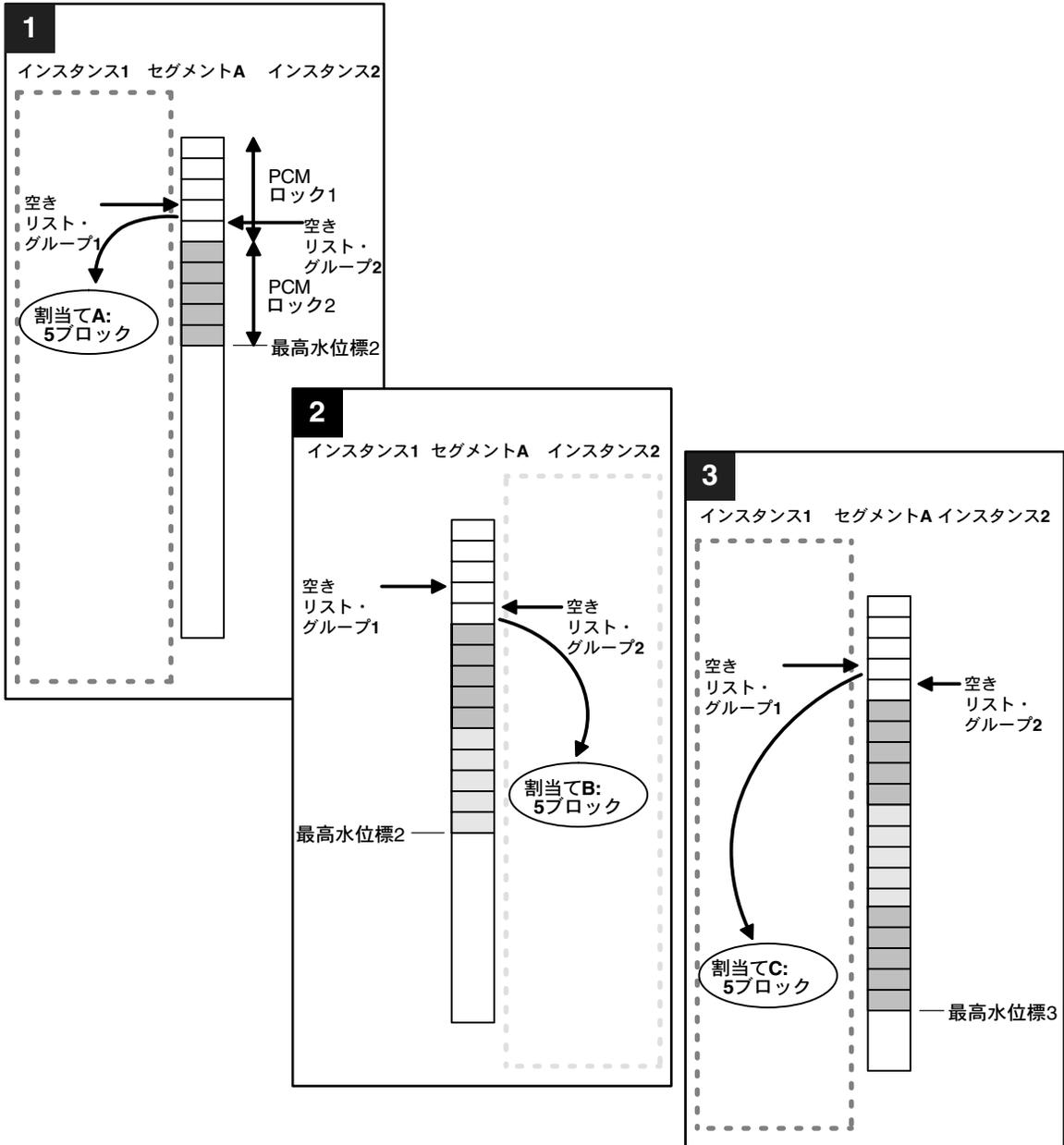
EACH オプションが指定されている場合、*file_list* 内の各ファイルには PCM ロックの #locks 番号が割り当てられます。各ファイル内では、各ブロックが処理する連続データ・ブロックの番号を !blocks によって指定します。

図 7-9 に、増分によるセグメント拡張の過程を示します。

- ステージ 1 では、インスタンス 1 がエクステントに 5 個のデータ・ブロックを割り当て、それをロック 2 が保護しています。
- ステージ 2 では、インスタンス 2 がさらに 5 個のデータ・ブロックを割り当て、それをロック 3 が保護しています。
- ステージ 3 では、インスタンス 1 がさらに 5 個のデータ・ブロックをもう一度割り当て、それをロック 4 が保護しています。

この方法で、インスタンス 1 のユーザー A がブロック 10 で作業している場合、どのインスタンスからもロック 2 によって処理されているブロック範囲内のブロックでは作業できません。これには、ブロック 6～10 が含まれます。

図 7-9 エクステント内のブロックの割当て



スケーラビリティおよび Oracle Parallel Server

この章では、Oracle Parallel Server のスケーラビリティ機能について説明します。この章の内容は次のとおりです。

- Oracle Parallel Server のスケーラビリティ機能
- パラレル処理が有利な場合
- マルチノード・パラレル実行
- クライアント対サーバー接続の概要
- スケーラビリティの4つのレベル

Oracle Parallel Server のスケーラビリティ機能

アプリケーションのパフォーマンスを拡張し、高可用性を維持するいくつかの機能を使用して、Oracle Parallel Server をインプリメントできます。次のような機能があります。

- 障害が発生しても、クライアントと Oracle Parallel Server データベース間で接続を持続します。
- 使用率が高い間は、複数のサーバーの接続を制御して、ノード間の作業負荷を均衡化します。
- 一般的なサービスの不具合の発生時に、アプリケーション対データベース接続の透過的なフェイルオーバーを提供する、いくつかのカスタマイズ可能なクライアント接続オプションを提供します。

この項では、これらの機能について説明します。

- [クライアント対サーバー接続の概要](#)
- [マルチスレッド・サーバーの使用によるスケーラビリティの拡張](#)
- [スケーラビリティの4つのレベル](#)

Oracle Database Configuration Assistant は、これらの機能の多くを自動的に構成します。これらの機能を手動で構成する方法は、以降の項に記載している箇所で『Oracle8i Net8 管理者ガイド』を参照してください。

拡張されたスループット：スケールアップ

お互いに独立して作業を実行できる場合、Oracle Parallel Server はこれらを異なるノードに分散できます。これによって、スケールアップを達成できます。つまり、同じ時間内により多くの処理をデータベースを介して実行できます。ただし、使用されるノードの数は、システムの目的によって異なります。

処理をより高速に実行できると、システムはより多くの作業を遂行します。たとえば、パラレル実行機能はスケールアップできます。つまり、問い合わせられたデータが 10 倍に増加した場合、またはより多くのユーザーに実行される場合にも、同じ応答時間を維持できます。パラレル実行機能のない Oracle Parallel Server もスケールアップできますが、これは、異なるノードで連続して同じ問合せを実行することによって行われます。

DSS、OLTP およびレポート・アプリケーションの作業負荷の混在に対しては、異なるノードで複数のプログラムを実行することによってスケールアップを実現できます。また、バッチ・プログラムをリライトし、多くのパラレル・ストリームに分割して複数の CPU を利用することによって、スピードアップもできます。

また、Oracle Parallel Server は、メモリー制限を克服することで改善された柔軟性も提供します。これによって、Oracle Parallel Server では何千ものユーザーに対応できます。必要に応じて、インスタンスを割当てまたは割当て解除できます。たとえば、データベースが増加を要求すると、より多くのインスタンスを一時的に割当てできます。これらのインスタンス

が必要とされなくなったら、そのインスタンスを割当て解除し、他の目的に使用できます。この機能は、インターネット・ベースのコンピューティング環境で有効です。

スピードアップおよびスケールアップ: パラレル処理の目標

次の2つの方法で、パラレル処理のパフォーマンスの目標を測定できます。

- スケールアップ
- スピードアップ

スケールアップ

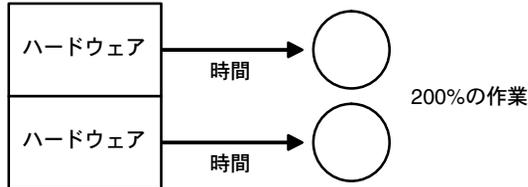
スケールアップとは、より大規模なシステムを使用することによって、同じ時間内にどれくらいの実行できるかを表す係数です。ハードウェアを追加することによって、スケールアップの計算式では時間を一定に保持し、実行可能なジョブの増加サイズを測定します。

図 8-1 スケールアップ

元のシステム:



パラレル・システム:



トランザクションの量が増加しても、スケールアップ率が良ければ、CPUなどのハードウェア・リソースを追加することで応答時間を一定に保持できます。

次の計算式を使用して、スケールアップを測定できます。

$$\text{スケールアップ} = \frac{\text{Volume_Parallel}}{\text{Volume_Original}}$$

各項目の内容は次のとおりです。

<i>Volume_Original</i>	小規模なシステム上で、一定の時間内に処理されたトランザクションの量。
<i>Volume_Parallel</i>	パラレル・システム上で、一定の時間内に処理されたトランザクションの量。

たとえば、元のシステムが、一定の時間内に 100 のトランザクションを処理し、パラレル・システムが、同じ時間内に 200 のトランザクションを処理する場合、スケールアップの値は 2 になります。2 の値は、理想的に比例したスケールアップを示します。つまり、ハードウェア数を 2 倍にすると、2 倍のデータ量を同じ時間内に処理できるということです。

スピードアップ

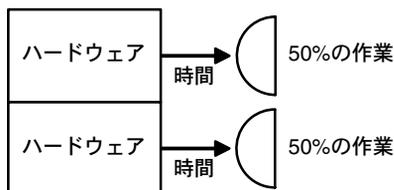
スピードアップとは、より多くのハードウェアを使用することによって、元のシステムよりも少ない時間で同じ作業をどの程度実行できるかということです。ハードウェアを追加することによって、スピードアップでは作業を一定に保持し、節約時間を測定します。図 8-2 に、各パラレル・ハードウェア・システムが、単一システム上で実行するのに必要な時間の半分で、元の作業の半分を実行する方法を示します。

図 8-2 スピードアップ

元のシステム:



パラレル・システム:



ただし、直線的なスピードアップにはならず、そのかわり、スピードアップは対数的である場合が多く発生します。たとえば、システムがサイズ x の作業を所要時間 n で実行できると仮定します。しかし、サイズ $2x$ の作業には、システムは $3n$ の所要時間を必要とすることがあります。

注意： ほとんどの OLTP アプリケーションでは、スケールアップのみが実現でき、スピードアップは期待できません。同期化によるオーバーヘッドは、実際には、スピードダウンの原因になります。

パラレル処理が有利な場合

この項では、Parallel Server から一般的に効果を得るアプリケーションについて説明します。

- 意思決定支援システム
- 異なるデータ・ブロックを更新するアプリケーション
- 部門別アプリケーション

意思決定支援システム

データの更新、挿入または削除をあまり行わないデータ・ウェアハウス・アプリケーションは、多くの場合 Oracle Parallel Server に対して適しています。問合せ処理集中型アプリケーション、および更新アクティビティが少ないアプリケーションは、非常に少ないオーバーヘッドで、異なるインスタンスを介してデータベースにアクセスできます。

データ・ブロックを変更しない場合、複数のノードが同じブロックをバッファ・キャッシュに読み込むことができ、追加の I/O またはロック・オペレーションなしで、そのブロックの問合せを実行できます。

意思決定支援アプリケーションは、データの変更がたまにしか行われないので、Oracle Parallel Server での使用には理想的です。たとえば、日中はほとんど問合せによってアクセスされ、オフピーク時に更新される財務トランザクションのデータベースなどに適しています。

異なるデータ・ブロックを更新するアプリケーション

異なるデータ・ブロックを更新する、または同じデータ・ブロックを異なる時間に更新するアプリケーションも、Parallel Server に適しています。たとえば、ユーザーがそれぞれ 1 つの表の集合を所有して使用するタイム・シェアリング環境などです。

バッファ・キャッシュに保持されるブロックの更新が必要なインスタンスは、これらのバッファを変更する間、1 つ以上のインスタンス・ロックを排他モードで保持する必要があります。インスタンス・ロックに対するこのような競合を削減するには、Parallel Server およびその Parallel Server 上で実行するアプリケーションをチューニングします。これを行うには、各インスタンスおよびアプリケーションによるデータの使用方法を計画し、それに応じて表をパーティション化します。

パーティション化データを処理する OLTP

異なるデータの集合を変更するオンライン・トランザクション処理アプリケーションは、Parallel Server の効果を最も得ます。たとえば、各支店（ノード）が各自の口座にアクセスし、ごくまれに他の支店の口座にアクセスするような、銀行支店システムなどがそうです。

大規模データベースへのランダム・アクセスを行う OLTP

ほとんどの場合、ランダムでデータベースにアクセスするアプリケーションも、Parallel Server の効果が得られます。これは、どのノードのバッファ・キャッシュよりもデータベースが非常に大きい場合にのみ当てはまります。たとえば、異なるノードから同時にアクセスされることがほとんどない個人記録を持つ、自動車部門のシステムなどがそうです。また、アーカイブされた税務記録や研究データ・システムなどもそうです。これらの例の場合、インスタンスがデータベースへの排他アクセスを行っても、多くのアクセスは I/O になります。1:1 ロックなどの Oracle の機能は、これらのアプリケーションのパフォーマンスを大幅に向上します。

部門別アプリケーション

部門別アプリケーションとは、ビジネスまたは部門ごとの処理に基づいて効率的にパーティション化されたアプリケーションです。このようなアプリケーションも、同じデータベース上の異なる表を主に変更するため、Oracle Parallel Server に適しています。たとえば、1 つのノードが在庫処理を、もう 1 つのノードが人事処理を、および別のノードが売上げ処理を

別々に実行するようなシステムなどです。この場合、管理するデータベースは3つではなく、1つのみです。

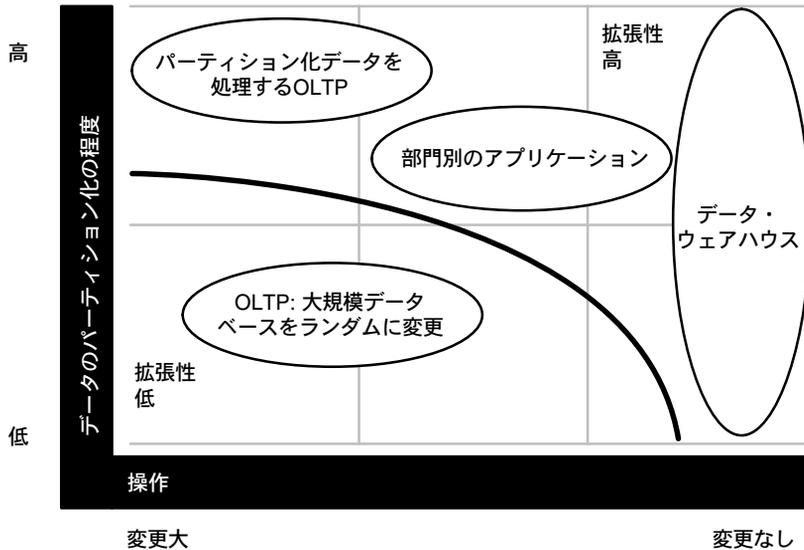
アプリケーション・プロファイル

オンライン・トランザクション処理（OLTP）アプリケーションは、対称型マルチプロセッサ（SMP）で最も効率的に実行します。これらのアプリケーションが適切にパーティション化できる場合は、クラスタおよび超並列（MPP）システム上でも効率的に実行されます。意思決定支援（DSS）アプリケーションは、SMP、クラスタおよび MPP システムでより効率的に実行されます。必要なアプリケーションに対して必要なパワーを提供する実装を選択してください。

図 8-3 に、異なるアプリケーション・タイプの相対的なスケーラビリティを示します。通常、最も右端の円で示されているデータ・ウェアハウス・アプリケーションは、更新があまり一般的ではなく、パーティション化の程度が他のアプリケーション・タイプよりも高いため、拡張性が高くなります。OLTP および部門別アプリケーションも、パーティション化および変更率の増加によって、拡張性が高くなります。

大規模データベースに対してランダムな変更を行う OLTP アプリケーションは、これまでは Parallel Server に適していないとみなされていました。ただし、このアプリケーションは、インターコネクトのような拡張ノード間通信チャネルを使用し、よりスケーラブルになってきています。たとえば、1つのインスタンスに対して表が変更され、その後もう1つのインスタンスがその表を読み込む場合は特にそうです。このような構成は、現在、以前のリリースに比べてよりスケーラブルになっています。

図 8-3 アプリケーションのスケラビリティ



マルチノード・パラレル実行

マルチノード・パラレル実行を使用して、パラレルに操作を実行できます。これによって、複雑な問合せおよび DML 文の処理時間を大幅に短縮できます。

クライアント対サーバー接続の概要

Oracle Parallel Server では、クライアント対サーバー接続は、いくつかのサブコンポーネントを使用して確立されます。クライアントは接続要求を、クラスタ内の接続先ノードに常駐するリスナー・プロセスに送ります。リスナー・プロセスとは、受信する接続要求を管理するプロセスです。リスナーは、接続オプションの構成方法に依存するいくつかの要因に基づいて、特定のノードを経由してクライアント対サーバー接続を許可します。

マルチスレッド・サーバーの使用によるスケラビリティの拡張

マルチスレッド・サーバーを使用すると、一定のメモリー量に対して、専用サーバーを使用した場合よりも多くのユーザーをサポートできるという点で、スケラビリティが拡張されます。ユーザーを追加するたびににかかるメモリーの増分コストは、専用サーバーを使用した場合よりも少なく済みます。ユーザー数のスケラビリティの向上に加えて、マルチスレッド・サーバーにはさらなるメリットがあります。

Oracle8i では、マルチスレッド・サーバーを必要とする機能が増えています。次のようなデータベース機能を使用する場合、マルチスレッド・サーバーが必要です。

- Oracle8i JServer
- 接続プーリング
- 接続集中化機能

参照： Oracle Parallel Server 環境用のマルチスレッド・サーバー構成については、『Oracle8i Net8 管理者ガイド』を参照してください。

サービス登録およびマルチスレッド・サーバーの機能

マルチスレッド・サーバーの重要な機能の1つに、「サービス登録」があります。この機能は、リスナーに、すべてのインスタンスおよび MTS ディスパッチャのカレント状態およびロード状態の他に、データベースのサービス名、インスタンス名およびネットワーク・アドレスを提供します。この情報によって、リスナーは、クライアント接続要求を適切なディスパッチャおよび専用サーバーに送信できます。

この情報はリスナーに登録されるため、データベース・サービスについてのこの静的情報を使用して listener.ora ファイルを構成する必要はありません。サービス登録は、マルチスレッド・サーバーの機能である接続時ロード・バランスの効果も拡張します。

接続時ロード・バランス

接続時ロード・バランス機能によって、複数のインスタンスおよびディスパッチャのある Oracle Parallel Server 環境では、非常に優れた効果を得られます。接続時ロード・バランスは、同じサービスに対する様々なインスタンスおよび MTS ディスパッチャ間のアクティブ接続の数を均衡化することによって、パフォーマンスを向上します。

リモート・リスナーに登録するサービス登録の機能によって、リスナーは常にすべてのインスタンスおよびディスパッチャを、その位置にかかわらず把握しています。このようにして、リスナーは、固有のサービスに対して受信したクライアント要求を、ロードが最も少ないインスタンスおよびロードが最も少ないディスパッチャに送信できます。

また、サービス登録は、MTS を使用してもしなくても、Connect-Time Failover およびクライアント・ロード・バランスを容易にします。

複数リスナーに対する Connect-Time Failover

サービス登録によって、リスナーは、インスタンスが起動しているかどうかを常に知ることができます。このため、複数のリスナーがサービスをサポートしている場合に、Connect-Time Failover を使用できます。これは、最初のリスナーに障害が発生した場合は、クライアント要求を異なるリスナーにフェイルオーバーするようにクライアントを構成することによって行います。この場合、クライアントがリスナーへの接続に成功するまで再接続が試行されます。インスタンスがダウンした場合、リスナーはネットワーク・エラーを戻します。

複数リスナーに対するクライアント・ロード・バランス

2つ以上のリスナーが1つのサービスをサポートする場合、クライアントは接続要求を様々なリスナーに対してランダムに送ることができます。ランダムな接続要求によって、負荷を分散し、単一リスナーへの過剰な負荷を回避できます。

ロード・バランスの他に、クライアントは、ランダムに選択されたリスナーに接続できない場合には、クライアントの接続要求を異なるリスナーに自動的にフェイルオーバーするように指定することもできます。指定されたリスナーが起動していないか、またはインスタンスが起動していないために接続が失敗することもあり、この場合、リスナーは接続を受け入れることができません。

参照： この項で説明した機能の構成については、『Oracle8i Net8 管理者ガイド』を参照してください。

スケーラビリティの4つのレベル

パラレル処理およびパラレル・データベースを正しく実装するためには、4つのレベルで最適化されたスケーラビリティが必要です。

- ハードウェアおよびネットワークのスケーラビリティ
- オペレーティング・システムのスケーラビリティ
- データベース管理システムのスケーラビリティ
- アプリケーションのスケーラビリティ

注意： 適切に設計されていないアプリケーションは、そのシステムで可能なスケーラビリティを十分に使用できない場合があります。同様に、アプリケーションが十分に拡張されていても、それを実行するハードウェアが拡張されていない場合は、期待するパフォーマンスは得られません。

ハードウェアおよびネットワークのスケーラビリティ

インターコネクトは、ハードウェアのスケーラビリティにとって重要です。それは、高速バスか低速なイーサネット接続かにかかわらず、すべてのシステムはCPUと接続するなんらかの方法を持つ必要があるためです。そして、インターコネクトの帯域幅および待ち時間が、ハードウェアのスケーラビリティを決定します。

帯域幅および待ち時間

インターコネクトのほとんどが、十分な帯域幅を持っています。帯域幅が大きい場合、事実上、待ち時間が長くなる場合があります。

ハードウェアのスケーラビリティは、待ち時間がどれだけ少ないかで決まります。ロック調整の通信量の通信は、LMD プロセス間での多くの非常に小さいメッセージで特徴付けられます。

たとえば高速道路で、100人の乗客を1台のバスで運ぶ場合と、別々の100台の車で運ぶ場合の違いを考えてみてください。後者の場合、効率は、車が高速道路にすばやく出入りできるかどうかという点に大きく依存します。たとえ、高速道路が5車線で多くの車が走れるようになっていても、入りランプが1車線しかなければ、それが高速道路に出るまでの障害になってしまいます。

パラレル実行など、ノード間の他の操作は帯域幅の大きさに依存しています。

ディスク入力および出力

ローカル I/O は、リモート I/O（ノード間で発生する I/O）よりも高速です。大量のリモート I/O が必要な場合、システムはスケーラビリティを失います。この場合、データをパーティション化し、データをローカルにします。

注意： 様々なクラスタ化の実装が、異なるハードウェア・ベンダーから入手できます。二重ポート・コントローラのある共有ディスク・クラスタでは、すべてのノードからの待ち時間は同じです。ただし、MPP（シェアード・ナッシング）システムでは、これは必ずしも当てはまりません。

オペレーティング・システムのスケーラビリティ

システムの最終的なスケーラビリティも、オペレーティング・システムのスケーラビリティに依存します。この項では、この要因の分析方法を説明します。

1つのノードが共有メモリー・システム（対称型マルチプロセッサ（SMP）の単一メモリーに複数CPUが接続されたシステム）である場合、ソフトウェアのスケーラビリティが重要な問題になります。オペレーティング・システムにおける同期化の方法によって、システムのスケーラビリティを判断できます。たとえば、非対称マルチプロセッシングでは、単一CPUのみがI/O割込みを処理できます。複数のユーザー・プロセスがオペレーティング・システムからリソースを要求するようなシステムを考えてみてください。

データベース管理システムのスケーラビリティ

Parallel Server アーキテクチャの重要な特徴は、内部と外部のパラレル化です。これは、スケーラビリティに大きな影響を及ぼします。主な違いは、オブジェクト・リレーショナル・データベース管理システム (ORDBMS) が問合せをパラレル化するか、外部プロセスが問合せをパラレル化するかという点です。

ノードがリモート・デバイスよりも主にローカル・デバイスにアクセスすることを確実にすることで、ディスク親和性によりパフォーマンスを向上できます。効率的な同期化メカニズムは、さらなるスピードアップおよびスケールアップを可能にします。

参照：

- 『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を参照してください。
- 『Oracle8i パフォーマンスのための設計およびチューニング』を参照してください。

アプリケーションのスケーラビリティ

アプリケーション設計は、システムの他の要素のスケーラビリティの効果を高めるために重要です。

注意： アプリケーションは、明確にスケーラブルになるように設計してください。

ハードウェア、ソフトウェアおよびデータベースがいかにスケーラブルでも、すべてのノードが更新する行を1つのみ持つ表は、1つのデータ・ブロックで同期化します。一意の順序番号を生成する処理を考えてみます。

```
UPDATE ORDER_NUM  
SET NEXT_ORDER_NUM = NEXT_ORDER_NUM + 1;  
COMMIT;
```

この順序番号を更新する必要があるすべてのノードは、この表の同じ行へアクセスするために待機する必要があります。この状況は、本質的にスケーラブルではありません。よりよいアプローチでは、順序を使用してスケーラビリティを改善します。

```
INSERT INTO ORDERS VALUES  
(order_sequence.nextval, ... )
```

この例では、順序番号を事前割当ておよびキャッシュして、スケーラビリティを改善できます。ただし、ビジネス・ルールによっては、拡張できないアプリケーションもあります。このような場合、そのルールのコストを判断する必要があります。

注意： クライアントは、サーバー・マシンにスケーラブルな方法で接続してください。つまり、ネットワークも、スケーラブルである必要があります。

参照： データベースの設計およびアプリケーションの分析については、『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を参照してください。

シーケンス・ジェネレータ

Oracle Parallel Server では、複数のインスタンス上のユーザーが、最小限の同期化で一意的順序番号を生成できます。

シーケンス・ジェネレータによって、複数のインスタンスが順序にアクセスし、増加できます。この場合、順序番号に対するインスタンス間の競合、およびトランザクションのコミットを待機する必要はありません。各インスタンスは独自の順序キャッシュを持つことができ、順序番号により高速にアクセスできます。分散ロック・マネージャのロックは、Oracle Parallel Server 内のインスタンス全体で、順序を調整します。

この項では、CREATE SEQUENCE 文およびそのオプションについて説明します。

- [CREATE SEQUENCE 文](#)
- [CACHE オプション](#)
- [ORDER オプション](#)

CREATE SEQUENCE 文

SQL 文 CREATE SEQUENCE は、同じシーケンス・ジェネレータにアクセスしたトランザクションを他のユーザーがコミットするのを待つことなく、複数のユーザーが一意的整数を生成できるデータベース・オブジェクトを設定します。

Oracle Parallel Server では、複数のインスタンス上のユーザーが、インスタンス間の協調または競合を最小限に抑えて一意的順序番号を生成できます。インスタンスのロックは、Oracle Parallel Server 内のインスタンス全体で、順序を調整します。

順序番号は、CYCLE オプションを使用する場合を除いて、常に一意です。ただし、ORDER オプションを使用せずに CACHE オプションを使用した場合、次の項で説明するように、順序番号を順不同に割当てできます。

参照： CREATE SEQUENCE および CYCLE オプションの詳細は、『Oracle8i SQL リファレンス』を参照してください。

CACHE オプション

CREATE SEQUENCE の CACHE オプションは、順序番号を事前割当てし、高速アクセス用にそれをインスタンスのシステム・グローバル・エリア (SGA) に保持します。また、キャッシュする順序番号の数を、引数として CACHE オプションに指定できます。デフォルト値は 20 です。

順序番号をキャッシュするとパフォーマンスが大幅に向上しますが、順序内のいくつかの番号が失われることがあります。順序番号を失うことは、主キーに対する一意の番号の生成のために順序を使用する場合など、アプリケーションによってはあまり重要ではありません。

ある順序に対するキャッシュは、その順序から番号が最初に要求されたときに移入されません。キャッシュされた番号の集合において最後の番号が割り当てられた後、キャッシュは別の番号の集合で再移入されます。

各インスタンスは、メモリー内の順序番号のキャッシュをそれぞれ保持します。1つのインスタンスが停止した場合、コミットされた DML 文で使用されたことのないキャッシュされた順序値は失われることがあります。失われる値の個数は、CACHE オプションの値に停止したインスタンスの数を掛けた値と同じくらい大きなものになることがあります。キャッシュされた順序番号は、インスタンスが正常に停止した場合でも失われることがあります。

ORDER オプション

CREATE SEQUENCE の ORDER オプションは、順序番号が要求された順序で生成されることを保証します。ORDER オプションは、タイムスタンプ番号、および複数のプロセスおよびインスタンスにまたがる要求の順序を示す必要があるその他の順序のために使用できません。

Oracle が順序番号を順序どおりに発行する必要がない場合、CREATE SEQUENCE の NOORDER オプションを使用すると、Oracle Parallel Server 環境で大幅にオーバーヘッドを削減できます。

注意： Oracle Parallel Server は、データベースがパラレル・モードでマウントされる場合、CREATE SEQUENCE に ORDER オプションがあるときは、CACHE オプションをサポートしません。Oracle は、各インスタンスがいくつかの順序値をキャッシュしている場合、順序付けを保証できません。したがって、CACHE オプションおよび ORDER オプションの両方を使用して順序を作成する必要がある場合、これらの順序は順序付けられませんが、キャッシュされません。

Oracle Parallel Server での Oracle パラレル実行

Oracle Parallel Server は、パラレル実行をノード間で操作するためのフレームワークを提供します。パラレル実行は、Oracle Parallel Server オプションの有無にかかわらず、同様に動作します。唯一の違いは、Oracle Parallel Server は、パラレル実行によって文の部分をノード間に分散することを可能にする点です。その結果、これらのノードは単一の間合せのように実行します。サーバーは、この文を共有ディスクに常駐する共通のデータベースに対して実行するさらに小さい操作に分割します。パラレル実行はサーバーが実行するため、このパラレル化は、外部 SQL レベルではなく、サーバー・オペレーションの低いレベルで行うことができます。

Oracle が文をパラレルで処理しない場合、Oracle は、ディスクを1つの I/O で逐次読み込みします。この場合、単一の CPU は表のすべての行をスキャンします。文をパラレル化すると、ディスクは複数の I/O でパラレルに読み込まれます。

複数の CPU は、表の部分をそれぞれパラレルにスキャンし、結果を集計できます。パラレル実行は、複数 CPU からのみでなく、より大きい I/O 帯域幅の可用性からも効果を得ます。

Oracle のパラレル実行は、Oracle Parallel Server の有無にかかわらず実行できます。Oracle Parallel Server オプションがない場合、パラレル実行はマルチノード・パラレル化を実行できません。Oracle Parallel Server は、パラレル・キャッシュ・アーキテクチャを使用して OLTP および DSS アプリケーションにおける共有メモリーのボトルネックを回避することによって、クラスタ化ハードウェアで実行される Oracle8i Enterprise Edition を最適化します。

高可用性および Oracle Parallel Server

この章では、Oracle Parallel Server を使用して高可用性を実装する最も実質的な方法と、その概念について説明します。この章の内容は次のとおりです。

- 高可用性の概念
- 高可用性の計画
- Oracle Parallel Server および高可用性
- 障害保護妥当性チェック
- フェイルオーバーおよび Oracle Parallel Server システム
- Oracle Parallel Server における高可用性の構成
- 高可用性の配置に向けて

高可用性の概念

常に可用性を提供するよう構成されたコンピュータ環境を、「高可用性」のシステムとといいます。そのようなシステムは、通常、障害が発生してもシステムを使用できるようにするための冗長ハードウェアおよび冗長ソフトウェアを備えています。適切に設計された高可用性のシステムでは、単一の致命的な障害箇所を持つことを回避できます。障害が発生する可能性のあるハードウェアまたはソフトウェアのコンポーネントはどれでも、同じタイプの冗長コンポーネントを備えています。

障害が発生すると、「フェイルオーバー」というプロセスにより、障害コンポーネントによって実行された処理をバックアップ・コンポーネントに移動します。フェイルオーバー・プロセスは、システム全体のリソースを再マスターし、部分トランザクションまたは障害トランザクションをリカバリし、システムを通常どおりに戻します。可能な場合、これらは100万分の数秒のうちに実行されます。フェイルオーバーがユーザーに対して透過的であるほど、システムの可用性は高いといえます。

Oracle Parallel Server は、高可用性のシステムです。第2章で説明するとおり、Oracle Parallel Server 環境の特徴であるクラスタは、予定された停止および予定外の停止の両方に際して連続的にサービスを提供できます。

可用性の評価

システムを分類し、システムの種類によって期待される可用性を評価できます。メール・サーバーやインターネット・サーバーなどのミッション・クリティカルおよびビジネス・クリティカルなアプリケーションには、使用頻度の低いアプリケーションの場合よりも、かなり優れた可用性が必要とされます。また、システムによっては1日24時間常に稼働が必要な場合もあれば、株式市場追跡システムなどのシステムでは、株式市場がオープンしているときなどの特定の時間には100%近い稼働が必要な場合もあります。

高可用性の基準

ソフトウェア業界では、一般的に、次の2種類の基準を使用して可用性を算出します。

- 平均リカバリ時間 (MTTR)
- 平均障害間隔時間 (MTBF)

ほとんどの障害の場合、業界では MTTR の問題に集中し、これらを減らすためにいかにシステムを最適に設計するかを調査します。MTBF は、一般的に、ハードウェアの可用性の基準として使用されます（この章では、MTBF の詳細は説明しません）。ただし、単一の致命的な障害箇所を回避するように Oracle Parallel Server クラスタを設計すると、コンポーネントに障害が発生しても、アプリケーションは必ずしも使用不可能にはなりません。したがって、Oracle Parallel Server は、アプリケーションの可用性の見地から、MTBF を大幅に減少させることができます。

一般的に使用されるもう1つの基準は「9の数」です。たとえば、システムが使用不可能な時間が1年あたり526分の場合には99.9%、つまり「9が3つの可用性」で、システムが使用不可能な時間が1年あたり5分の場合には99.999%、つまり「9が5つの可用性」です。

アプリケーション環境の管理、方法論の検証および管理手順の変更に対する基本原則および厳密なプロセスを説明せずに、「9の数による可用性」を検討することは困難です。そのため、ここでは、Oracle Parallel Serverが障害時にどれだけMTTRを減少させることができるかについて集中的に説明します。このことは、すなわち、システム全体の「9の可用性」がより向上することにつながります。

停止の原因

停止時間は、次の2つのカテゴリに分類できます。

- 予定された停止時間
- 予定外の停止時間

予定された停止時間

予定されたメンテナンス、製品のアップグレードおよびアプリケーションの変更は、通常、これらの時間内で実行されます。エンド・ユーザーは、一般的に、これらの期間はシステムにアクセスできません。Oracle Parallel Serverには、定型のメンテナンスを実行するために必要な予定停止時間を最小化するための数多くの機能が提供されています。これらの機能には、システム・メンテナンス時の別のノードへのフェイルオーバー、オンライン再編成、オンライン・バックアップ、パーティション化操作などがあります。

予定外の停止時間

予定外の停止時間とは、コンポーネント障害またはシステム障害（あるいはその両方）が「システムの停止」という結果を招いた場合です。ユーザーのエラーもまた、予定外の停止時間を生み出すことがあります。そのような障害は、ハードウェアまたはソフトウェアの問題であったり、CPU、メモリー、オペレーティング・システム、データベースまたはネットワークに関する問題であったりします。

すでに述べたように、適切に設計された Oracle Parallel Server システムは、ほとんどの障害からシステムを保護し、単一の致命的な障害箇所のない環境を提供する冗長コンポーネントを備えています。ハードウェア・ベンダーとの共同作業が、Oracle Parallel Server に完全な冗長クラスタ環境を構築するためのキーです。

高可用性の計画

高可用性は、十分な計画および慎重なシステム設計の結果得られます。高可用性は、次の2つのレベルで計画できます。

- 広い観点を持ったシステム・レベル
- 障害の原因となり得る多数の項目のリストを確認する障害保護レベル

システム・レベルの計画

システム・レベルの計画には次のものがあります。

- 容量計画
- 冗長性計画

容量計画

高可用性において、システムが完全に使用可能とみなされるには、トランザクション処理が適切に行われることが必要です。この章では容量拡張計画については説明していませんが、アプリケーションの拡張を管理するための適切なシステム・リソースは、可用性を維持するために重要です。

単一インスタンスの Oracle を使用した、単一の対称型マルチプロセッシング (SMP) ・マシン上でアプリケーションを実行した場合、一般的な拡張方法は、このデータベースをより大きな SMP マシンに移行することです。ただし、ご使用のハードウェア・ベンダーの製品ラインによっては、このオプションがない場合があります。

Oracle Parallel Server では、システムにノードを追加し、容量を増加してアプリケーションの拡張に対処できます。これは、データベースを停止することなく、また、既存のクライアントのトランザクションへの影響も最小限に抑えながら、オンラインで処理できます。

冗長性計画

冗長性計画とは、単一のコンポーネント障害がシステム可用性を減少させないようにシステム・コンポーネントを二重化することです。冗長コンポーネントは、障害から保護するためにハイエンド SMP マシンでよく使用されます。たとえば、冗長電源供給および冗長冷却ファンは、ハイエンド SMP サーバー・システムではめずらしくありません。クラスタ化された Oracle Parallel Server 環境は、単一の致命的な障害箇所のない完全な冗長性を作成することによって、この冗長アーキテクチャを高いレベルに拡張できます。

注意： 高可用性のシステムをインストールする場合、ハードウェアおよびソフトウェアが1つの単位として認証されていることを確認してください。

Oracle Parallel Server および高可用性

Oracle Parallel Server は、Oracle の標準機能の上に高レベルの可用性を構築します。ファースト・スタート・リカバリやオンライン再編成などの、すべての単一インスタンスの高可用性機能は、Oracle Parallel Server にも適用されます。ファースト・スタート・リカバリは、オンライン・アプリケーションのパフォーマンスへの影響を最小限に抑えながら、MTTR を大幅に減少できます。オンライン再編成によって、予定された停止時間の期間を短縮できます。従来はメンテナンス時にオフラインで実行していた操作が、基礎となるオブジェクトをユーザーが更新している最中でも、オンラインで実行できます。Oracle Parallel Server には、これらすべての Oracle 標準機能が備わっています。

すべての Oracle の標準機能に加えて、Oracle Parallel Server では、クラスタ化によって提供される冗長性を利用して、N ノード・クラスタ内の N-1 ノード障害に対しても可用性を提供できます。つまり、すべてのユーザーは、クラスタ内に1つでも使用可能なノードがあれば、すべてのデータにアクセスできます。

Oracle Parallel Server を高可用性に構成するには、次の項で説明するような、クラスタのハードウェアおよびソフトウェアのコンポーネントに関する問題を考慮する必要があります。

クラスタ・コンポーネントおよび高可用性

この項では、次のクラスタ・コンポーネントに関する高可用性の問題について説明します。

- [クラスタ・ノード](#)
- [クラスタ・インターコネクト](#)
- [記憶デバイス](#)
- [オペレーティング・システム・ソフトウェアおよび Cluster Manager](#)
- [データベース・ソフトウェア](#)

参照： これらのコンポーネントの詳細は、[第2章](#)を参照してください。

クラスタ・ノード

すでに述べたように、Oracle Parallel Server 環境は、完全に冗長であり、データベースを構成するすべてのディスクにすべてのノードがアクセスできます。1つのノードでの障害は、別のノードがトランザクションを処理する能力に影響を及ぼしません。クラスタに1つの障害のないノードがある限り、すべてのデータベース・クライアントは、すべてのトランザクションを処理できます。ただし、その1つのノードの容量制限によって応答時間は増加します。

クラスタ・インターコネクト

インターコネクト冗長性は、クラスタ化システムの中でよく見落とされます。これは、平均障害時間 (MTTF) が一般的に数年で、そのため、クラスタ・インターコネクト冗長性の優先順位が高くないためです。また、システムおよび洗練のレベルによって、冗長クラスタ・インターコネクトはコストが高くなったり、ビジネスへの調整が十分でないことがあります。

ただし、冗長クラスタ・インターコネクトは、完全冗長クラスタの重要な側面です。これがなければ、システムは単一の致命的な障害箇所がまったくないとはいいきれません。クラスタ・インターコネクトは、様々な理由で障害が発生することがあり、それらのすべての原因が明らかになっていないわけではありません。また、MTTFが製造元から提供されている場合にも、完全とはいえません。インターコネクトは、切替えインターコネクトの発振器の故障などのデバイスの機能不全、または人為的なエラーによって障害が発生する場合があるためです。

記憶デバイス

Oracle Parallel Server は、単一のデータ・イメージを操作します。クラスタ内のすべてのノードは、同じデータ・ファイルにアクセスします。データベース管理者は、ハードウェア・ベースのミラー化を使用して、冗長メディアをメンテナンスすることをお勧めします。これに関しては、Oracle Parallel Server は単一インスタンスの Oracle と同様です。ディスク冗長性は、RAID など、基礎となるハードウェアおよびソフトウェアのミラー化に依存します。

オペレーティング・システム・ソフトウェアおよび Cluster Manager

すでに述べたように、Oracle Parallel Server 環境には完全なノード冗長性が備わっています。各ノードは、それぞれのオペレーティング・システム・コピーを実行します。したがって、ノード冗長性に関する同じ考慮点が、オペレーティング・システムにも適用できます。Cluster Manager は、オペレーティング・システムの拡張です。Cluster Manager ソフトウェアも、クラスタのすべてのノードにインストールされるため、完全冗長性が保証されます。

データベース・ソフトウェア

Oracle Parallel Server では、データベース・バイナリは各ノードのローカル・ディスクにインストールされ、クラスタの各ノードで1つのインスタンスが実行されます。すべてのイン

スタンスはすべてのデータに同様にアクセスでき、どのようなトランザクションでも処理できます。このように、Oracle Parallel Server では、完全なデータベース・ソフトウェア冗長性が保証されています。

災害時計画

Oracle Parallel Server は、基本的に単一サイトの、高可用性のソリューションです。つまり、クラスタ内のノードは、一般的に、同じ部屋ではないとしても、同じビルの中に存在します。そのため、災害時計画は重要です。災害時計画とは、火災、洪水、台風、地震、テロ行為などに対する計画です。システムがどれほどミッション・クリティカルであるか、また、システムのある場所がそれらの災害に対してどのようになる傾向があるかによって、災害時計画は、高可用性の重要なコンポーネントとなることがあります。

Oracle では、スタンバイ・データベースおよびレプリケーションなどのその他のソリューションを提供し、より広範囲な災害リカバリ計画を促進します。これらのソリューションは、1つのクラスタがプライマリ・データベースのホストとして機能し、別のリモート・システムまたはクラスタが災害リカバリ・データベースのホストとして機能する Oracle Parallel Server で使用できます。Oracle Parallel Server は、どちらのサイトにも必要ではありません。

障害保護妥当性チェック

システム・レベルの問題を慎重に検討した後、Oracle Parallel Server 環境を、予想される障害から保護できるかどうかの妥当性チェックを行います。次に示すものは、このプロセスに使用できる障害原因です。これらは広範囲ですが、すべての原因を網羅しているわけではありません。

- クラスタ・コンポーネント
- CPU
- メモリー
- インターコネクト・ソフトウェア
- オペレーティング・システム
- Cluster Manager
- Oracle データベース・インスタンス・メディア
- 破損 / 消失した制御ファイル
- 破損 / 消失したログ・ファイル
- 破損 / 消失したデータ・ファイル人為エラー
- 削除されたデータベース・オブジェクト

「システム・レベルの計画」で説明するとおり、Oracle Parallel Server 環境は、クラスタ・コンポーネント障害およびソフトウェア障害から保護されます。ただし、メディア障害および人為エラーは、システム停止時間の原因となることがあります。Oracle Parallel Server は、単一インスタンスの Oracle の場合と同様、1つのファイル集合を操作します。このため、メディア障害を回避するため、効果的な対策を採用する必要があります。

RAID ベースの冗長性の実行は、ファイルの消失は回避できますが、ファイルの破損は回避できない場合があります。Oracle Parallel Server 環境で誤ってデータベース・オブジェクトを削除した場合、単一インスタンスのデータベースでの場合と同じ方法でそのオブジェクトをリカバリできます。これらは、Oracle Parallel Server システムの主な制限です。これらの制限をのぞけば、Oracle Parallel Server システムは非常に堅牢で高可用性のシステムです。

システムを配置した後、重要な問題は、フェイルオーバーの透過性およびその期間です。次の項では、フェイルオーバーを詳細に説明します。

フェイルオーバーおよび Oracle Parallel Server システム

次の項では、フェイルオーバーの基本、およびそれを高可用性のシステムにインプリメントするために Oracle Parallel Server が提供している様々な機能について説明します。内容は次のとおりです。

- フェイルオーバーの基本
- クライアント・フェイルオーバー
- サーバー・フェイルオーバー

フェイルオーバーの基本

フェイルオーバーには、高可用性のシステムに適切なインスタンス監視機能またはハートビート・メカニズムが備わっていることが必要です。通常操作に対する機能に加えて、システムはフェイルオーバーの間、すばやく正確にリソースを同期化する必要があります。

同期化のプロセス、または再マスター化には、システムにマスター化されたリソースの制御を的確に想定すると同時に、障害システムの適切な停止が必要です。また、的確な再マスター化には、システムがクラスタ間のリソースについての的確な情報を持っていることも必要です。つまり、システムは、ローカルと同様、リモート・ノードにもリソース情報を記録する必要があります。これによって、フェイルオーバーおよびリカバリに必要な情報が、リカバリするインスタンスに使用可能になります。

フェイルオーバーの期間

フェイルオーバーの期間には、システムがシステム全体のリソースを再マスターし、障害からリカバリするのに必要な時間が含まれます。フェイルオーバー・プロセスの存続期間は、保証されたプラットフォーム上では比較的短い期間となる場合があります。すでに処理を行っているユーザーの場合、フェイルオーバーは、サーバー・フェイルオーバー処理およびクライアント・フェイルオーバー処理の両方を意味します。新しく処理を開始するユーザーの場合、フェイルオーバーはサーバー・フェイルオーバー時間のみを意味します。

クライアント・フェイルオーバー

データベース・クライアント接続からシステム障害を隠すことは重要です。そのような接続には、クライアント・サーバー環境のアプリケーション・ユーザー、またはマルチティア・アプリケーション環境のミドルティア・データベース・クライアントが含まれます。データベース障害が発生した場合、クライアントには接続の消失を気付かせないようにする必要があります。適切に構成されたフェイルオーバー・メカニズムは、クライアント・セッションの経路をクラスター内の使用可能なノードに透過的に変更します。Oracle データベースのこの機能を「透過的アプリケーション・フェイルオーバー」といいます。

透過的アプリケーション・フェイルオーバーの概念

透過的アプリケーション・フェイルオーバー（TAF）では、接続が切断された場合、アプリケーション・ユーザーはデータベースへ自動的に再接続できます。アクティブ・トランザクションはロール・バックしますが、別のノード経由で作成された新しいデータベース接続は、元のものと同様です。これは、接続がどのような原因で切断された場合でも同様です。

Oracle Parallel Server での透過的アプリケーション・フェイルオーバーの動作方法

透過的アプリケーション・フェイルオーバーを使用すると、アプリケーションに使用できるインスタンスが1つでも残っていれば、クライアントは接続の切断に気付きません。DBA が、どのアプリケーションをどのインスタンス上で実行するかを制御し、各アプリケーションにフェイルオーバー順序を作成します。

TAF から影響を受けるアクティブ・データベース接続の要素

通常のクライアント / サーバー・データベース操作の間、クライアントはクライアントとサーバーが通信できるようにデータベースへの接続を維持します。サーバーに障害が発生すると、接続にも障害が発生します。クライアントが次にその接続を使用しようとする時、エラーが発生します。この場合、ユーザーはデータベースにログインし直す必要があります。

ただし、透過的アプリケーション・フェイルオーバーがあれば、Oracle はデータベースへの新しい接続を自動的に取得します。これによって、ユーザーは、元の接続に障害がなかったかのように作業を継続できます。

次に、アクティブ・データベース接続に対応付けられた要素を示します。

- クライアント / サーバー・データベース接続
- コマンドの実行におけるユーザーのデータベース・セッション
- フェッチに使用されるオープン・カーソル
- アクティブ・トランザクション
- サーバー側プログラム変数

透過的アプリケーション・フェイルオーバーでは、これらの要素が自動的にリストアされます。ただし、その他の要素については、透過的アプリケーション・フェイルオーバーが接続をリカバリするように、アプリケーション・コードに埋め込まれている必要がある場合があります。

参照： TAF の構成方法については、『Oracle8i Net8 管理者ガイド』を参照してください。

透過的アプリケーション・フェイルオーバーの使用

システム障害時にクライアント・セッションをフェイル・オーバーすることが透過的アプリケーション・フェイルオーバーの強力なメリットですが、透過的アプリケーション・フェイルオーバーには、システムの可用性も向上させるというメリットがあります。それらのメリットは次のとおりです。

- [トランザクション停止](#)
- [ロード・バランス](#)

トランザクション停止

メンテナンスまたは修正（あるいはその両方）のため、ノードをサービスから外すことも、時には必要です。たとえば、アプリケーション・クライアントへのサービスに割り込まずに、パッチ・リリースを適用する必要がある場合があります。SHUTDOWN 文の TRANSACTIONAL 句を使用することによって、既存のすべてのトランザクションが完了するまで停止イベントが遅延するように、ノードをサービスから外すことができます。このように、クライアント・セッションは、トランザクションの境界で、クラスタの別のノードへ移行することができます。

また、トランザクション停止を実行した後、発行された新しいトランザクションはクラスタ内の別のノードに転送されます。既存のトランザクションがすべて完了したとき、ノードで SHUTDOWN IMMEDIATE が実行されます。

ロード・バランス

データベースは、トランザクションを適時に処理した場合に使用可能になります。ロードがノードの容量を超えた場合、クライアント・トランザクション応答時間は、悪影響を受け、データベースの可用性は損なわれます。そこで、アプリケーションの可用性の応答時間を維

持するために、クライアント・セッション・グループをロードの少ないノードに手動で移行できることが重要となります。

透過的アプリケーション・フェイルオーバーの制限事項

接続が切断された場合、次のような影響が現れます。

- サーバー上のすべての PL/SQL パッケージの状態は、フェイルオーバー時に消失します。
- ALTER SESSION 文の効果が消失します。
- トランザクション処理中にフェイルオーバーが発生した場合、ユーザーが OCITransRollback コールを発行するまで、後続の各コールはエラー・メッセージの原因となります。その後、Oracle は OCI 成功メッセージを発行します。このメッセージをチェックして、追加の操作が必要かどうか確認してください。
- フェイルオーバー状態のカーソルで作業を継続すると、エラー・メッセージの原因となります。
- フェイルオーバー後の最初のコマンドが SQL SELECT 文または OCIStmtFetch 文ではない場合、エラー・メッセージが表示されます。
- フェイルオーバーは、OCI リリース 8.0 以降を使用してプログラムされたアプリケーションにのみ効力があります。

フェイルオーバー中にデータベース・クライアントに起きること

フェイルオーバー処理中の重要な問題は、障害が既存のクライアント接続からマスクされるエクステンツがあることです。

問合せクライアント フェイルオーバーの際は、処理中の問合せは再発行され、最初から処理されます。これによって、元の間合せに長時間かかった場合に、次の問合せの所要時間が延長される場合があります。透過的アプリケーション・フェイルオーバーでは、障害は問合せクライアントに対して通知されません。クライアントが認識できるのは、応答時間の増加のみです。クライアントの間合せが、クライアントが再接続する存続ノードのバッファ・キャッシュ内のデータで対応できるものであれば、応答時間の増加は最小で済みます。透過的アプリケーション・フェイルオーバーで PRECONNECT メソッドを使用した場合、存続インスタンスに再接続するための時間を節約することによって、さらに応答時間を短縮できます。

クライアント間合せが、再接続したノードのバッファ・キャッシュのデータで対応できない場合、クライアント間合せを処理するためにディスク I/O が必要です。ただし、データ・ファイルへのアクセスは、サーバー側がリカバリしなければ許可されません。サーバー側のリカバリがまだ完了していなければ、クライアント・トランザクションでは、サーバー側のリカバリが完了するまでシステムが休止されます。

コールバック関数を使用して、クライアントがディレイを障害と誤解しないように、クライアントにフェイルオーバーを通知することもできます。これによって、クライアントが接続を手動で再開するのを回避できます。

DML クライアント コーディングされたアプリケーションが Oracle コール・インタフェース (OCI) ・ライブラリを完全に利用すると想定すると、INSERT、UPDATE および DELETE 操作を実行する DML データベース・クライアントの場合、障害インスタンスの実行中の DML トランザクションは、クライアントが認識しないまま存続インスタンス上で再起動されます。これによって、手動で再接続することなくアプリケーション・フェイルオーバーを実行できますが、アプリケーション・レベルのコーディングが必要です。必要なコーディングは、基本的に、特定の Oracle エラー・コードを処理し、これらのエラー・コードが戻されたときに再接続を実行するものです。

アプリケーション・コーディングが配置されていない場合、障害インスタンスの INSERT、UPDATE および DELETE の各操作は、未処理の Oracle エラー・コードを戻します。トランザクションを再発行して実行する必要があります。トランザクションを再発行すると、Oracle は、クライアント接続を存続インスタンスに転送します。クライアント・トランザクションでは、サーバー側のリカバリが完了するまでシステムが休止されます。

サーバー・フェイルオーバー

Oracle Parallel Server のサーバー側のフェイルオーバーは、多くのサーバー・プラットフォームで使用可能な、通常の、ホストベースのフェイルオーバー・ソリューションとは異なります。

ホストベース・フェイルオーバー

多くのオペレーティング・システム・ベンダー、およびその他のクラスタ・ソフトウェア・ベンダーは、高可用性のアプリケーション・フェイルオーバー製品を提供しています。これらのフェイルオーバー・ソリューションは、特定のプライマリ・クラスタ・ノード上のアプリケーション・サービスを監視します。そして、それらのサービスを、必要に応じて、セカンダリ・クラスタ・ノードにフェイルオーバーします。ホストベースのフェイルオーバー・ソリューションには、一般的に、任意のデータベース・アプリケーションのために有用な作業を実行する 1 つのアクティブ・インスタンスがあります。セカンダリ・ノードがプライマリ・ノードのアプリケーション・サービスを監視し、プライマリ・ノードのサービスが使用できない場合にフェイルオーバーを開始します。

ホストベース・システムのフェイルオーバーには、通常、次のステップがあります。

1. ハートビートを監視することによって障害を検出します。
2. Cluster Manager でクラスタ・メンバーシップを再編成します。
3. ディスクのオーナーシップをプライマリ・ノードからセカンダリ・ノードに転送します。

4. アプリケーションおよびデータベース・バイナリを再起動します。
5. アプリケーションおよびデータベースのリカバリを実行します。
6. フェイルオーバー・ノードへのクライアント接続を再確立します。

Oracle Parallel Server フェイルオーバー

Oracle Parallel Server では、非常に高速のサーバー側フェイルオーバーが提供されます。これは、Oracle Parallel Server の同時実行（アクティブ-アクティブ・アーキテクチャ）によって達成されます。つまり、複数の Oracle インスタンスが複数ノードで同時にアクティブで、同じデータベースへのアクセスを同期化します。すべてのノードは同時実行オーナーシップを持ち、すべてのディスクにアクセスします。1つのノードで障害があった場合、クラスタ内のすべての他のノードは、すべてのディスクへのアクセスを維持します。転送するためのディスク・オーナーシップはなく、データベース・アプリケーション・バイナリはすでにメモリーにロードされています。

データベースのサイズによって、フェイルオーバーの期間は異なります。データベースが大きいほど、または、そのデータ・ファイルのサイズが大きいほど、Oracle Parallel Server を使用することに対するフェイルオーバーの効果も大きくなります。これは、ホストベースのフェイルオーバー環境におけるプライマリ・インスタンスからセカンダリ・インスタンスへのディスク・オーナーシップの転送は、フェイルオーバーされる必要のあるファイルのサイズおよび数に比例するためです。ホストベースのフェイルオーバー環境で、アプリケーション・バイナリおよびデータベース・バイナリを再起動するための追加コストは、固定コストで、アプリケーション・バイナリおよびデータベース・バイナリのサイズ、およびアプリケーション初期化処理のエクステントに比例します。

障害シナリオの最中のクライアント・フェイルオーバーの動作については、すでにクライアント・フェイルオーバーの項で分析したとおりです。新しいクライアント接続がクラスタ内の使用可能なノードに転送され、障害ノードの特定の既存クライアント接続が透過的にフェイルオーバーするよう構成されます。ただし、データベース・クライアントが使用可能なノードでトランザクションを処理し始めるには、Oracle Parallel Server がサーバー側のリカバリ処理を完了している必要があります。

Oracle Parallel Server のフェイルオーバーの動作方法

Oracle Parallel Server で、障害ノードに必要なリカバリ処理には、次のようなものがあります。

- 障害の検出
- クラスタ・メンバーシップの再編成
- データベース・リカバリの実行

障害の検出

Oracle Parallel Server は、障害検出を Cluster Manager ソフトウェアに依存します。これは、Cluster Manager はハートビート機能をメンテナンスするためです。Cluster Manager が、動作していないノードを検出するのに要する時間は、構成可能なハートビート・タイムアウト・パラメータに関係します。この値は、ほとんどのシステムで構成できます。デフォルト値は通常、1 分間です。タイムアウトの設定が低すぎる場合、クラスタが、一時的な障害のためノードに障害が発生したと不適切に判断する場合がありますため、この値は、不適切な警告または不適切な障害検出の数に影響を及ぼします。障害が検出されると、クラスタの再編成が行われます。

クラスタ・メンバーシップの再編成

ノードに障害が発生した場合、Oracle はそのクラスタ・メンバーシップ状態を変更する必要があります。これを「クラスタ再編成」といいます。これは、通常、高速で行われます。その期間は、クラスタ内の存続ノードの数に比例します。Oracle Parallel Server は、この情報を Cluster Manager ソフトウェアに依存します。

Oracle Parallel Server の分散ロック・マネージャは、ソフトウェアに Cluster Manager インタフェースを提供し、ノードがクラスタに追加または削除されたときに、Oracle インスタンスにクラスタ・メンバーシップ・マップを公開します。クラスタ・ノード上の分散ロック・マネージャの LMON プロセスは、各ノードの Cluster Manager と通信し、その情報をそれぞれの Oracle インスタンスに公開します。

LMON には、別の有用な機能もあります。ノードがクラスタのメンバーでなくなった場合、障害のないノードは、クラスタ内に、共有ディスクへのメッセージや書込みなど、そのノードの影響を受けません。LMON によって提供されるこれらのサービスを「クラスタ・グループ・サービス (CGS)」ともいいます。障害によってクラスタ内のノードのメンバーシップ状態に変更が生じた場合、LMON が、PCM ロックの再マスターやインスタンス・リカバリなどのリカバリ処理を開始します。

このとき、Oracle Parallel Server 環境はシステム休止の状態で、ほとんどのクライアント・トランザクションは、必要なリカバリ処理が完了するまで中断されます。

データベース・リカバリの実行

Oracle Parallel Server で、データベース・リカバリに必要なステップは、次のとおりです。

- 障害インスタンスの PCM ロック・リソースの再マスター化
- キャッシュ・リカバリおよびトランザクション・リカバリを含むインスタンス・リカバリ

インスタンスに障害が発生した場合、障害インスタンスの PCM ロック・リソースは存続クラスタ・ノード上で再マスターされる必要があります。これを「分散ロック・マネージャのデータベース再構築」ともいいます。

障害インスタンスの PCM ロック・リソースの再マスター化

ロックの再マスター化に要する時間は、ロック・データベース内の PCM ロックの数の関数です。この数は、バッファ・キャッシュのサイズに依存します。1:1 解放可能ロックの場合、バッファ・キャッシュ内の各個別データベース・ロックには 1 つの PCM ロックが必要です。ただし、解放可能ロックの場合、ロック・リソースはすでに解放されていて、再マスターの必要がない場合もあります。1:N 固定ロックの場合、各ロックが複数のブロックをロックするため、ロックの数は初期化パラメータによって決定されます。

この段階で、すべてのロック情報は破棄され、存続している各インスタンスは、障害が発生したときに保持していたすべてのロックを再取得します。ロック領域は、残りのインスタンス間に均一に分散されます。どのようなロック要求の場合でも、要求がローカルで満たされる可能性は $1/n$ で、要求がリモート操作を含む可能性は $(N-1)/n$ です。存続インスタンスが 1 つの場合は、すべてのロック操作はローカルで実行されます。

障害インスタンス PCM ロックの再マスター化が完了した後、障害インスタンスの実行中のトランザクションはクリーン・アップされる必要があります。これを「インスタンス・リカバリ」といいます。

インスタンス・リカバリ

インスタンス・リカバリが実行されるには、Oracle Parallel Server のアクティブ・インスタンスが、障害を検出し、障害が発生した Oracle Parallel Server のインスタンスのかわりに必要なリカバリ処理を実行することが必要です。LMON プロセスによって、障害を検出した最初の Oracle Parallel Server のインスタンスは、障害インスタンスの REDO ログ・ファイルを引き継ぐことによって障害インスタンスのリカバリを制御し、インスタンス・リカバリ処理を実行します。このため、共有ロー論理ボリュームまたはクラスタ・ファイル・システムなどの共有デバイスに、REDO ログ・ファイルが必要です。

キャッシュ・リカバリ、つまり、障害インスタンスのオンライン REDO ログ・ファイルが再実行された場合、およびトランザクション・リカバリ、つまり、障害インスタンスでコミットされていないすべてのトランザクションがロールバックされた場合に、インスタンス・リカバリは完了したとみなされます。トランザクション・リカバリは遅延のかたちで実行されるため、クライアント・トランザクションは、キャッシュ・リカバリが完了したときに処理を開始できます。

キャッシュ・リカバリ

キャッシュ・リカバリが実行されるには、Oracle が、障害インスタンスのオンライン REDO ログを再実行する必要があります。Oracle は、キャッシュ・リカバリをパラレルで実行します。つまり、障害が発生した Oracle インスタンスの REDO ログを再実行するために、パラレルのスレッド処理を開始します。REDO ログの再実行に要する時間の長さを、予測可能な時間にしておくことが重要である場合があります。Oracle8 のファースト・スタート・リカバリ機能は、この機能を備えています。

ファースト・スタート・リカバリでは、FAST_START_IO_TARGET というパラメータを使用して、インスタンス・リカバリに必要な REDO ログ再実行の量全体にきめ細かいコントロールを行います。REDO ログ再実行は、一定サイズの REDO ログの最後尾をメンテナンスする連続的チェックポイント・メカニズムによって実行されます。データベース・クライアントではシステムが短時間休止されているだけのため、サーバー側のリカバリ処理に必ずしも気付くわけではありません。FAST_START_IO_TARGET に特定の値を設定することは、システム障害時のシステム休止の時間を許容範囲内に保つためには重要です。

Oracle では、非ブロック化ロールバック機能を提供しています。そのため、オンライン・ログ・ファイルが再実行されるとすぐに、完全なデータベース・アクセスを開始できます。キャッシュ・リカバリが完了すると、Oracle はトランザクション・リカバリを開始します。

トランザクション・リカバリ

トランザクション・リカバリは、障害インスタンスのコミットされていないすべてのトランザクションのロールバックで構成されています。これらは、コミットされていない処理中トランザクションであり、Oracle はこれらをロールバックする必要があります。

Oracle8 ファースト・スタート・ロールバックは、遅延プロセスとしてバックグラウンドでロールバックを実行します。Oracle は、複数バージョン読取り一貫性テクノロジーを使用して、停止したトランザクションによってブロックされた行のみのオンデマンド・ロールバックを提供します。そのため、新しいトランザクションは、最小の遅延で実行されます。新しいトランザクションは、停止したトランザクション全体がロールバックされるのを待つ必要がないため、長時間実行トランザクションがデータベース・リカバリに要する時間に影響を及ぼすことはありません。

Oracle8 ファースト・スタート・ロールバックは、多くのリカバリ・プロセスを起動するリカバリ・コーディネータを使用して、停止したトランザクションをパラレルでロールバックします。単一インスタンスの Oracle は、1 つのノードの CPU を使用して停止したトランザクションをロールバックします。

Oracle Parallel Server には、クラスタを意識したファースト・スタート・ロールバック機能があります。これは、クラスタのすべての CPU ノードを使用してパラレル・ロールバック操作を実行するものです。各クラスタ・ノードは、リカバリ・コーディネータおよびリカバリ・プロセスを起動して、パラレル・ロールバック操作を支援します。データベースがパラレル・ロールバック操作を行うには、すべてのクラスタ・リソースを認識し、利用するため、ファースト・スタート・ロールバック機能は、このようにクラスタを意識したものとなっています。

デフォルトの動作はトランザクション・リカバリを遅延させますが、クライアント・トランザクションの処理を許可する前に、トランザクション・リカバリを完了させるようにシステムを構成するように選択することができます。この場合、複数ノード間でトランザクション・リカバリを平行化する Oracle Parallel Server の機能は、その有効性がユーザーにとってさらに明確になります。

Oracle Parallel Server における高可用性の構成

次の項では、Oracle Parallel Server で提供されている次の 3 つの高可用性の構成について説明します。

- デフォルト N ノード Parallel Server 構成
- 高可用性の基本構成
- 高可用性の共有ノード構成

デフォルト N ノード Parallel Server 構成

デフォルト N ノード Oracle Parallel Server 構成は、デフォルトの Oracle Parallel Server 環境です。クライアント・トランザクションは、クラスタのすべてのノードで処理され、クライアント・セッションは、接続時にロード・バランス化されます。高可用性の環境を作成するためにクラスタ・ノード間にロードを分散させることによって、応答時間は、CPU やメモリーなどの使用可能なクラスタ・リソースに最大限に利用されます。

N ノード Oracle Parallel Server 構成の利点

ノード障害が発生した場合、すでに述べたように、別のノードの Oracle Parallel Server インスタンスが必要なリカバリ処理を実行します。障害インスタンス上のデータベース・クライアントは、クラスタの存続 (N-1) インスタンス間にロード・バランス化される場合があります。各存続インスタンス上で増えるロードは最小に保持され、応答時間を許容範囲内に維持することによって可用性は増加します。この構成では、データベース・アプリケーションの作業負荷は、すべてのノード間に分散され、その結果、クラスタ・マシン・リソースの高い使用率が提供されます。

高可用性の基本構成

Oracle Parallel Server を高可用性の基本構成に構成するのは簡単です。高可用性の基本構成とは、1つのノードのプライマリ・インスタンスがユーザー接続を受け入れ、別のノードのセカンダリ・インスタンスは、プライマリ・ノードに障害が発生した場合にのみ接続を受け入れます。この構成は、特定のインスタンスへのトランザクションのルーティングを手動で制御することによって可能ですが、Oracle Parallel Server では、これを行うために、プライマリ / セカンダリ・インスタンス機能を提供しています。

プライマリ / セカンダリ・インスタンス機能は、`init.ora` ファイルのパラメータ `ACTIVE_INSTANCE_COUNT` を 1 に設定することによって構成します。最初にデータベースをマウントするインスタンスがプライマリ・インスタンスの役割を担うことになります。他のインスタンスがセカンダリ・インスタンスとなります。プライマリ・インスタンスに障害が発生した場合、セカンダリ・インスタンスがプライマリ・インスタンスの役割を担います。障害インスタンスがアクティブ状態に戻ったとき、それはセカンダリ・インスタンスになります。

セカンダリ・インスタンスは、Cluster Manager がプライマリ・インスタンスの障害を通知した後、分散ロック・マネージャの再構成、キャッシュ・リカバリおよびトランザクション・リカバリが開始される前にプライマリ・インスタンスになります。存続インスタンスへのリダイレクションは透過的に実行され、アプリケーションのプログラミングは必要ありません。クライアント接続文字列への若干の構成変更のみが必要です。

プライマリ / セカンダリ・インスタンス構成では、すべての N ノード Oracle Parallel Server 環境のように、両インスタンスが同時に実行されます。ただし、データベース・アプリケーション・ユーザーは、指定されたプライマリ・インスタンスにのみ接続できます。プライマリ・ノードはすべての分散ロック・マネージャのロックをマスター化します。これによって、ノード間の通信を最小化し、標準の、単一ノード・データベースに匹敵するパフォーマンス・レベルが提供されます。

セカンダリ・インスタンスは、「リモート・クライアント」という特別に構成されたクライアントによって、バッチ問合せレポート操作またはデータベース管理作業に利用されることがあります。これによって、セカンダリ・ノードはある程度使用可能になります。このことはまた、プライマリ・インスタンスから CPU 容量へのロードを軽減し、冗長ノードに対する投資の正当性を証明します。

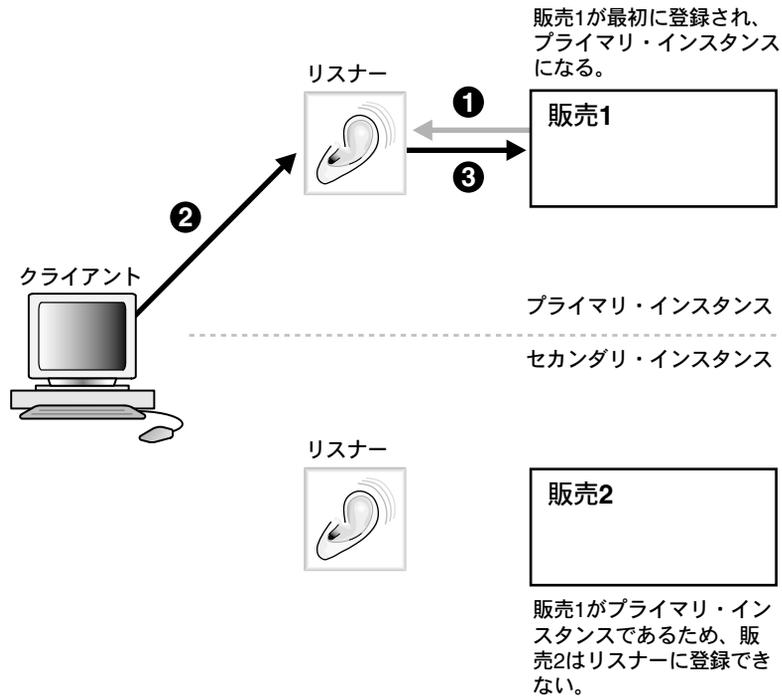
参照： クライアント接続文字列の構成についての情報は、『Oracle8i Parallel Server セットアップおよび構成ガイド』を参照してください。

プライマリ / セカンダリ・インスタンス機能は、専用サーバー環境およびマルチスレッド・サーバー環境の両方で動作します。ただし、以降の項で説明するように、それぞれで異なった働きをします。

専用サーバー環境のプライマリ/セカンダリ・インスタンス

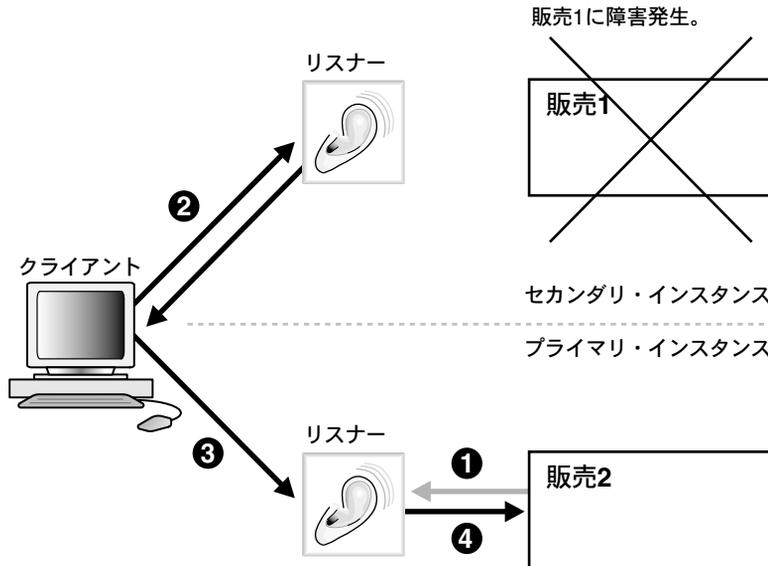
図 9-1 で示すように、専用サーバー環境では、インスタンス間のリスナー登録はありません。そのため、特定のインスタンスのリスナーへの接続要求は、そのインスタンスのサービスへのみ接続されます。この動作は、専用サーバー環境のデフォルトの N ノード Oracle Parallel Server クラスタに似ています。

図 9-1 専用サーバー環境のプライマリ/セカンダリ・インスタンス機能



プライマリ・インスタンスに障害が発生した場合、クライアントからの再接続要求は障害インスタンスのリスナーによって拒否されます。セカンダリ・インスタンスはリカバリを実行し、プライマリ・インスタンスになります。クライアント要求を再発行すると、クライアントは新しいプライマリ・インスタンスのリスナーを使用して接続を再構築します。その後、新しいプライマリ・インスタンスのリスナーは、クライアントを新しいプライマリ・インスタンスに接続します。

図 9-2 専用サーバー環境のノード障害

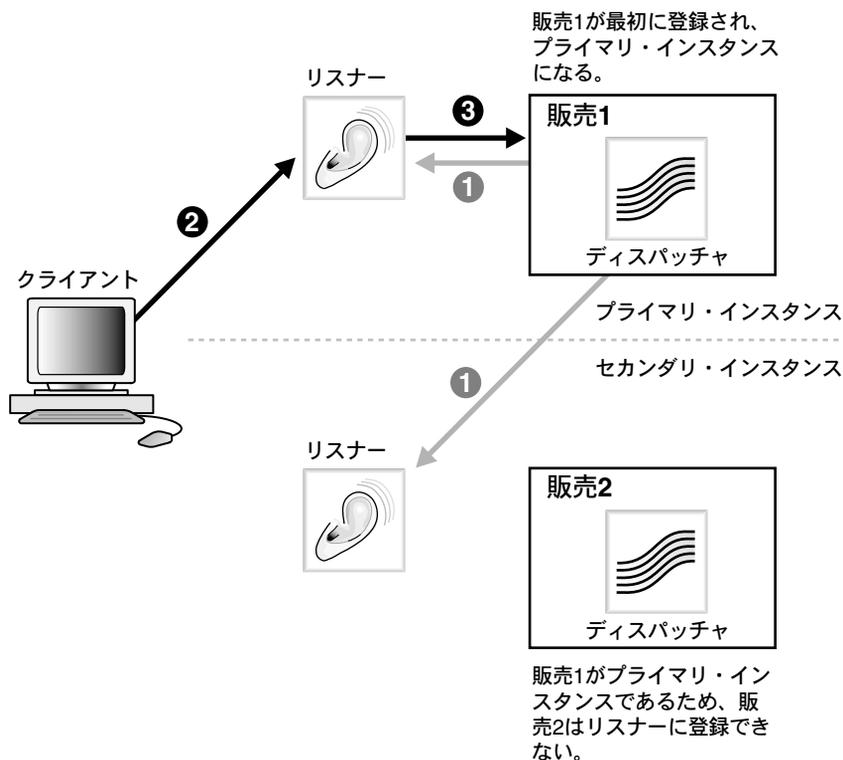


プライマリ / セカンダリ・インスタンスおよびマルチスレッド・サーバー

Oracle Parallel Server では、マルチスレッド・サーバー・モードで実行した場合、再接続パフォーマンスの効果が得られます。これは、クラスタ内のすべてのディスパッチャおよびリスナーのクロス登録によって行われます。

プライマリ / セカンダリ構成では、図のステップ 1 のように、プライマリ・インスタンスのディスパッチャのみがクラスタ内のすべてのリスナーを認識します。クライアントは、ステップ 2 のように、いずれかのリスナーに接続します。ステップ 2 は、プライマリ・ノード・リスナーへの接続のみ示しています。関連リスナーは、ステップ 3 のように、クライアントをディスパッチャに接続します。ステップ 3 はプライマリ・ノードのリスナー / ディスパッチャ接続のみ示しています。

図 9-3 マルチスレッド・サーバー環境のプライマリ / セカンダリ・インスタンス機能

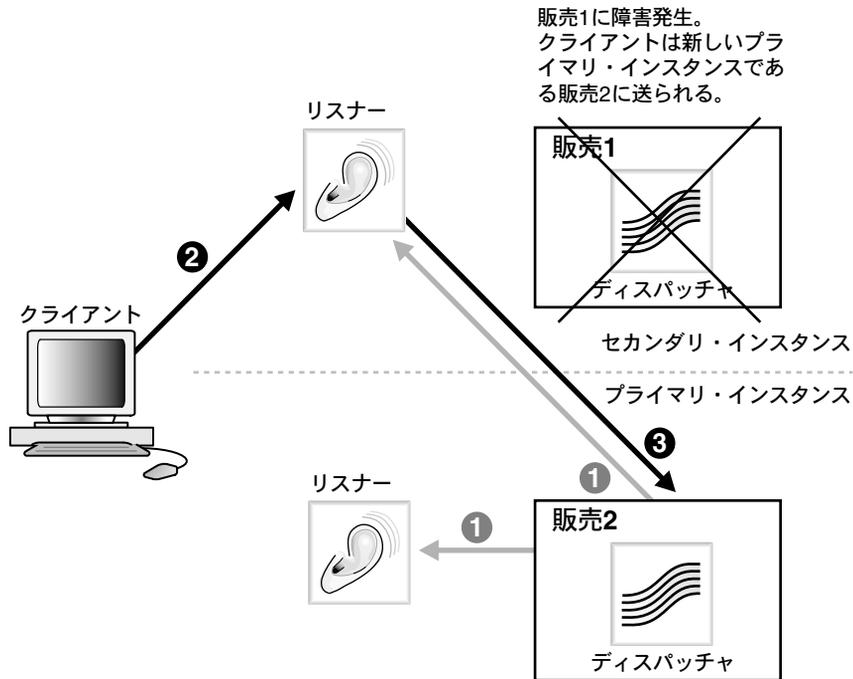


特別に構成されたクライアントは、セカンダリ・インスタンスを使用してバッチ操作ができます。たとえば、バッチ・レポート作業、索引作成操作は、セカンダリ・インスタンス上で実行できます。

参照： セカンダリ・インスタンスへの接続方法は、『Oracle8i Parallel Server セットアップおよび構成ガイド』を参照してください。

図 9-4 では、プライマリ・ノードに障害が発生した場合、ステップ 1 のように、セカンダリ・インスタンスのディスパッチャがリスナーに登録します。クライアントがいずれかのリスナーを介してデータベースへの再接続を要求した場合、リスナーはその要求をセカンダリ・インスタンスのディスパッチャへ送ります。

図 9-4 マルチスレッド・サーバー環境のノード障害



高可用性の基本構成の利点

デフォルトの2ノード構成ではなくこの構成を使用するには、2つの理由があります。プライマリ / セカンダリ・インスタンス機能は、次のことを提供します。

- Nノード構成を得るための移行パス
- 1つのノードを超えて拡張しないアプリケーションに対する、高可用性のソリューション

Nノード構成への移行パス

プライマリ / セカンダリ構成を使用するのは、アプリケーション環境を Oracle Parallel Server 環境に移行するための段階的な方法です。すべてのクライアント・トランザクションは、指定された時間に1つのノード上のみで実行されるため、Oracle Parallel Server のチューニング問題は最小限で済みます。同時に2つ以上のノードをチューニングするのに対して、1度に1つのノードをチューニングするだけで済むため、システム問題のトラブルシューティングも簡単になります。

可用性も、データベース管理者の環境の管理、チューニングおよびトラブルシューティング能力に依存します。Oracle Parallel Server のプライマリ / セカンダリ構成を行うと、データベース管理スタッフが Oracle Parallel Server を段階的に使用しやすくなります。

拡張しないアプリケーションに対する可用性のソリューション

アプリケーションは、いくつかの理由で単一インスタンスを超えて拡張できません。その最も一般的な理由は、アプリケーション・レベルのシリアライズ化点です。アプリケーション設計によって、単一のアプリケーション・リソース上にボトルネックが発生する場合、アプリケーションはそのリソースの容量を超えて拡張できません。

アプリケーションが拡張できない理由は他にもあります。たとえば、更新集中型アプリケーションおよび非パーティション化アプリケーションは、Oracle Parallel Server のディスクベースの同期化メカニズムのために適切に拡張できない場合があります。そのような場合は、同期化またはブロック ping のコストが高くなる場合があります。ただし、Oracle8i キャッシュ・フュージョン・テクノロジーへの拡張を行うと、この問題はあまり重要でなくなります。

前述の2つのカテゴリのいずれかに適応するユーザー環境は、Oracle Parallel Server のプライマリ / セカンダリ構成に適しています。

高可用性の共有ノード構成

N ノード構成で Oracle Parallel Server を実行すると、最大限にクラスタ・リソースを利用できます。ただし、すでに説明したとおり、これは常に可能または推奨できるとは限りません。一方、フェイルオーバーに備えてアイドル・ノードを持つために必要な投資は、多くの場合、高い費用がかかります。このような状況の場合、高可用性の共有ノード構成が適しています。

このタイプの構成は、通常、いくつかのノードを持ち、それぞれのノードが個別のアプリケーション・モジュール、またはすべてのアプリケーション・サービスが1つの Oracle Parallel Server データベースを共有する場合は、サービスを実行します。個別の指定ノードをフェイルオーバー・ノードとして設定できます。Oracle Parallel Server インスタンスがそのノードで実行している間は、通常の操作中はどのユーザーもそのノードに送られません。アプリケーション・ノードのいずれかに障害が発生した場合、高可用性のノードに作業負荷を送ることができます。

この構成は別々のノードで実行される必要のあるアプリケーションを検討するときに有効ですが、ミドルティア・アプリケーションまたはトランザクション処理モニターが適切なアプリケーション・ユーザーを適切なノードに送ることができる場合に、最も有効です。プライマリ/セカンダリ構成とは異なり、高可用性のノードへの作業負荷の移行を自動化するデータベース設定はありません。アプリケーション、つまりミドルティア・ソフトウェアは、障害アプリケーション・ノードのユーザーを指定された高可用性のノードに送る役目を担う必要があります。また、アプリケーションは、障害ノードが操作可能になった場合に、ユーザーのフェイル・バックを制御する必要もあります。フェイル・バックは、処理中のユーザー作業のあるフェイルオーバー・ノードを、引き続いて起こるノード障害から解放します。

高可用性の共有ノード構成の利点

この構成では、アプリケーション・パフォーマンスはフェイルオーバー時にメンテナンスされません。N ノード・クラスタ構成では、同じ作業負荷がより小さいクラスタ・ノード集合に再分散されるため、1/N の割合でアプリケーション・パフォーマンスが低下します。

高可用性の配置に向けて

クラスタ化システム上の Oracle Parallel Server は、非常に障害対応性である完全冗長環境を提供します。この高可用性モデルの中心は、Oracle Parallel Server アーキテクチャで、すべてのクラスタ・ノードには、すべてのデータに同等のアクセスを持つアクティブ・インスタンスがあります。いずれかのノードに障害が発生した場合、すべてのユーザーは、他のクラスタ・ノードの障害のないインスタンスを経由する方法で、すべてのデータにアクセスできます。障害ノードの実行中のトランザクションは、障害を検出した最初のノードによってリカバリされます。このように、Oracle Parallel Server を使用すると、エンド・ユーザー・アプリケーションの可用性への中断を最小限に抑えられます。

第IV部

参照情報

第IV部に含まれる参照情報は、次のとおりです。

- [付録A「リリース間の相違点」](#)
- [付録B「制限事項」](#)

リリース間の相違点

この付録では、Oracle Parallel Server におけるリリース間の相違点を説明します。

- リリース 8.1 とリリース 8.1.6 の相違点
- リリース 8.0.4 とリリース 8.1 の相違点
- リリース 8.0.3 とリリース 8.0.4 の相違点
- リリース 7.3 とリリース 8.0.3 の相違点
- リリース 7.2 とリリース 7.3 の相違点
- リリース 7.1 とリリース 7.2 の相違点
- リリース 7.0 とリリース 7.1 の相違点
- バージョン 6 とリリース 7.0 の相違点

参照： ご使用のデータベースのアップグレードについては、『Oracle8i 移行ガイド』を参照してください。

リリース 8.1 とリリース 8.1.6 の相違点

新機能

- プライマリ / セカンダリ・インスタンス - プライマリ / セカンダリ・インスタンス機能を使用して、高可用性の基本構成を実装できます。この機能は、2 ノード Oracle Parallel Server 環境で機能します。1 つのノード上のプライマリ・インスタンスはユーザー接続を受け入れ、もう 1 つのノード上のセカンダリ・インスタンスはプライマリ・ノードに障害が発生した場合にのみ接続を受け入れます。

廃止されたパラメータ

LM_PROCS パラメータは廃止されました。

廃止された統計情報

次の統計情報は廃止されました。

- global cache consistent read from disk
- global cache fairness down converts

新しい統計情報

新しい統計情報は次のとおりです。

- global cache cr block send time - ブロックの送信にかかる合計時間
- global cache cr block log flushes - ログ・フラッシュの回数
- global cache cr block log flush time - ログ・フラッシュにかかる合計時間
- global cache prepare failures - 準備中に障害が発生した回数

デフォルトのパラメータ設定の変更

GC_ROLLBACK_LOCKS のデフォルトの設定は「0-128=32!8REACH」です。これは、0 ~ 129 のロールバック・セグメントをロックで保護します。

Oracle によって自動的に設定される LM_LOCKS および LM_RESS

Oracle は、初期化パラメータ・ファイルの設定に基づいて、LM_LOCKS および LM_RESS の値を自動的に設定します。

リリース 8.0.4 とリリース 8.1 の相違点

キャッシュ・フュージョン・アーキテクチャの変更

あるインスタンスが、別のインスタンスが保持しているブロックの一貫読込み (CR) を要求すると、キャッシュ・フュージョン処理が、要求されたブロックの CR コピーを、要求側インスタンスにインターコネクト経由で直接送ります。このことによって、読込み / 書込み競合中の、インスタンス間のキャッシュ一貫性競合が大幅に減少します。

キャッシュ・フュージョンを実装するには、いくつかのバックグラウンド・プロセスおよびフォアグラウンド・プロセス (つまり、LMON および LCK) が、インターコネクトを介して、あるインスタンスから別のインスタンスへ直接通信する必要があります。新しいプロセスであるブロック・サーバー・プロセス (BSP) は、コミットされていないトランザクションをロールバックし、要求側インスタンスに送信用の CR サーバー・ブロックをコピーします。このことによって、キャッシュの一貫性を維持するのに必要な ping が減少し、パフォーマンスが大幅に向上します。

キャッシュ・フュージョンによって、OLTP およびハイブリッド・アプリケーションへの Oracle Parallel Server の配置がしやすくなります。これまで、ランダムに変更されるデータベースは、Parallel Server には適さないと考えられてきました。キャッシュ・フュージョンおよび高度なインスタンス間インターコネクト・テクノロジーの出現によって、OLTP およびハイブリッド・アプリケーションはよりスケーラブルになってきています。たとえば、表が、あるインスタンスで変更され、別のインスタンスがその表を読み込む場合などは、特にそうです。

新しいビュー

新しいビューは次のとおりです。

- V\$DLM_ALL_LOCKS は、ブロックする側のロックかブロックされる側のロック、また他のロックのタイプなどの、ロックに関する統計情報を示します。
- V\$DLM_RESS は、ロックに対応付けられたすべてのリソースを、ロック・タイプに従って示します。
- V\$DLM_CONVERT_LOCAL は、ローカル・ノードでオープンされているロックのロック変換統計情報を示します。
- V\$DLM_CONVERT_REMOTE は、リモート・ノードでオープンされているロックのロック変換統計情報を示します。
- V\$DLM_MISC は、DML メッセージ情報を示します。

GMS の削除

リリース 8.1 では、GMS (グループ・メンバーシップ・サービス) の機能が、GMS モジュールからベンダー固有の Cluster Manager (CM) および Oracle データベース・カーネルへ移動されました。リリース 8.1 では、個々の GMS モジュールを Oracle ユーザーが見ることはできません。

この変更によって、ベンダーのハードウェアと Oracle との互換性が大幅に改善されました。ユーザーにとっては、この変更によって、CM の使用およびメンテナンスが簡単になりました。CM は、インスタンス起動時に自動的に起動するようになり、メンバー・サービスを、手動で起動および停止する必要がなくなりました。

パラレル・トランザクション・リカバリから「ファースト・スタート・パラレル・ロールバック」への改名

「パラレル・トランザクション・リカバリ」機能は、「ファースト・スタート・パラレル・ロールバック」と改名されました。名前の変更の他に、リリース 8.0 では、SMON はロールバック・セグメント・リカバリを逐次処理しており、これによって、ロールバック・リカバリ期間が長くなっていました。リリース 8.1 では、ファースト・スタート・パラレル・ロールバックによってリカバリ時間が短縮され、そのため、データベースがより早く使用可能になります。パラレル・ロールバックは、パラメータ `FAST_START_PARALLEL_ROLLBACK` (以前の `PARALLEL_TRANSACTION_RECOVERY`) の値が 1 より大きい場合に、複数のプロセスを使用してロールバック・セグメントをリカバリします。

このパラメータのデフォルトは `LOW` です。これは、パラレル・リカバリがパラレル・リカバリを実行するために、SMON の他に `CPU_COUNT` の 2 倍以上のプロセスを使用しないことを意味します。

より正確な設定を判断するには、2 つの新しい表である `V$FAST_START_SERVERS` および `V$FAST_START_TRANSACTIONS` の内容を調べてください。また、トランザクションをリカバリするのに必要な平均時間および希望するリカバリ期間を考慮してください。

`FAST_START_PARALLEL_ROLLBACK` を 1 より大きい値に設定すると、SMON はマルチ・リカバリ・プロセスを起動して、ロールバック・ファイルのリカバリされていないロールバック・セグメントの量を処理します。SMON が起動するプロセスの数は、`FAST_START_PARALLEL_ROLLBACK` の値によって制限されています。

インスタンス登録への変更

サービスを識別するために以前使用されていた単一の名前 (SID) が、3 つのレベルの指定に置き換えられました。インスタンス登録の新しいパラメータは次のとおりです。

<code>SERVICE_NAME</code>	サービスの最高レベルのビュー名で、 <code>TNSNAMES.ORA</code> で指定します。インスタンスまたはノードにまたがってもかまいません。
---------------------------	---

SERVICE_NAMES	サービスのインスタンス名であり、複数のノードにまたがることができます。このパラメータは INIT.ORA で指定します。
INSTANCE_NAME	サービスの中間レベル層の名前。インスタンスの ORACLE_SID に対応します。

クライアントは、必要なハンドラまたはインスタンスを指定しなくてもサービスに接続できます。したがって、自動ロード・バランスによって、最適なハンドラが最適なインスタンスに選択できます。次に、ロード・バランスについて説明します。

リスナーのロード・バランス

TNS リスナーは、複数ノードにまたがる分散サービスを介してロード・バランスを実行するようになりました。サービス名、インスタンス名およびハンドラ名を使用して、ロード・バランス動作を決定します。

1. クライアント・プログラムが、そのプログラムが接続するサービス名を指定します。
2. リスナーが、サービス内でロードが最も少ないインスタンスを検索します。
3. リスナーが、インスタンス内でロードが最も少ないハンドラを検索します。
4. リスナーが、クライアントを最適なハンドラにリダイレクトします。

診断プログラムの拡張

Oradebug は、実行時に問題のあるシステムを診断および解決するために、オラクル社のコンサルティングおよびサポート要員が使用するユーティリティです。Oradebug の機能は、Oracle Parallel Server 用に拡張されています。

Oracle Parallel Server Management (OPSM)

OPSM は、Parallel Server の管理を簡単にするオプションです。リリース 8.1 での OPSM の拡張によって、すべてのプラットフォーム上で Parallel Server を管理するための単一の汎用インタフェースを提供します。

OPSM の詳細は、『Oracle Parallel Server Management ユーザーズ・ガイド』を参照してください。

Parallel Server のインストーलおよびデータベース構成

Oracle Universal Installer および Oracle Database Configuration Assistant の両方とも、クラスタを認識します。リリース 8.1 では、Oracle Parallel Server のインストールに必要な Installer セッションは、1 つのみです。Installer はノード情報をユーザーから収集し、必要な Oracle 製品を指定されたノードに配布します。その後、Oracle Parallel Server Assistant を起動してインスタンスを設定し、データベースを作成します。

Oracle Parallel Server Assistant がこの処理を終了すると、Parallel Server はすべてのノードで使用可能になり、Parallel Server の構成情報が保存されます。OPSM はこの情報を使用して新しい Parallel Server を管理できます。

ジョブとインスタンスの親和性

ジョブとインスタンスの親和性とは、インスタンスに対するジョブの対応付けです。新しい DBMS_JOB パッケージを使用すると、特定のインスタンスまたは任意のインスタンスが、ユーザー発行のジョブを Oracle Parallel Server 環境で実行できるかどうかを指定できます。

リリース 8.1 のこの機能を使用して、ロード・バランスを改善し、ブロックの ping を制限します。たとえば、Oracle Parallel Server およびレプリケーションを同時に使用すると、クラスタ環境のすべてのインスタンスが遅延トランザクション・キューからトランザクションを伝播する場合、遅延トランザクション・キューでブロックの ping の問題が発生する場合があります。表に対するアクティビティを Parallel Server のクラスタ内の 1 つのインスタンスのみに制限することによって、ping を制限できます。詳細は、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』を参照してください。

廃止されたパラメータ

次のパラメータは、リリース 8.1 で廃止されました。

- GC_LCK_PROCS
- GC_LATCHES
- PARALLEL_DEFAULT_MAX_INSTANCES
- LOG_FILES
- OPS_ADMIN_GROUP
- CACHE_SIZE_THRESHOLD
- OGMS_HOME
- ALLOW_PARTIAL_SN_RESULTS
- SEQUENCE_CACHE_ENTRIES

リリース 8.0.3 とリリース 8.0.4 の相違点

新しい初期化パラメータ

次の初期化パラメータが Oracle Parallel Server に追加されました。

- OGMS_HOME
- GC_LATCHES
- PARALLEL_SERVER

廃止された初期化パラメータ

次の初期化パラメータが廃止されました。

- MTS_LISTENER_ADDRESS
- MTS_MULTIPLE_LISTENERS

廃止された起動パラメータ

- PARALLEL
- EXCLUSIVE

動的パフォーマンス・ビュー

次のビューが変更されました。

- V\$DLM_LOCKS

グループ・メンバーシップ・サービス

新しいオプションが OGMSCTL コマンドに追加されました。

リリース 7.3 とリリース 8.0.3 の相違点

新しい初期化パラメータ

次のパラメータが Oracle Parallel Server に追加されました。

- FREEZE_DB_FOR_FAST_INSTANCE_RECOVERY
- LM_LOCKS
- LM_PROCS
- LM_RESS
- INSTANCE_GROUPS
- PARALLEL_INSTANCE_GROUP
- OPS_ADMIN_GROUP
- ALLOW_PARTIAL_SN_RESULTS

廃止された GC_* パラメータ

次のグローバル・キャッシュ・ロック初期化パラメータが廃止されました。

- GC_DB_LOCKS パラメータ
- GC_FREELIST_GROUPS パラメータ
- GC_ROLLBACK_SEGMENTS パラメータ
- GC_SAVE_ROLLBACK_LOCKS パラメータ
- GC_SEGMENTS パラメータ
- GC_TABLESPACES パラメータ

変更された GC_* パラメータ

GC_* パラメータで設定した値は、素数に調整されるのではなく、入力したとおりの値になります。

次のパラメータが変更されました。

- GC_FILES_TO_LOCKS
- GC_ROLLBACK_LOCKS
- GC_RELEASABLE_LOCKS

動的パフォーマンス・ビュー

新しいビューは次のとおりです。

- V\$RESOURCE_LIMIT
- V\$DLM_CONVERT_LOCAL
- V\$DLM_CONVERT_REMOTE
- V\$DLM_LATCH
- V\$DLM_MISC
- V\$FILE_PING
- V\$CLASS_PING

次のビューは変更されました。

- V\$BH
- V\$SESSIONS
- V\$SYSSTAT

グローバル動的パフォーマンス・ビュー

V\$ROLLNAME を除くそれぞれの V\$ ビューに対応する、グローバル動的パフォーマンス・ビュー（GV\$ 固定ビュー）が追加されました。

分散ロック・マネージャ

Oracle Parallel Server リリース 8.0 は、外部の分散ロック・マネージャに依存しません。ロック管理機能は現在 Oracle 内部にあります。統合分散ロック・マネージャは、外部の Node Monitor に依存します。

LMON および LMD n プロセスが追加されました。

インスタンス・グループ

論理的にインスタンスをグループ化し、関連するすべてのインスタンスをまとめて操作する機能が追加されました。

グループ・メンバーシップ・サービス

グループ・メンバーシップ・サービス (GMS) は、ロック・マネージャ (LM) およびその他の Oracle コンポーネントが、インスタンス間の初期化および調整を行うために使用します。

ファイングレイイン・ロック

Oracle Parallel Server リリース 8.0 では、ファイングレイイン・ロックがすべてのプラットフォームで使用可能です。これはデフォルトで使用可能です。

クライアント側のアプリケーション・フェイルオーバー

Oracle8 では、データベースへの接続が中断された場合に、アプリケーションが自動的に再接続する機能をサポートしています。

Recovery Manager

メディア障害からのリカバリの手段として、Recovery Manager (RMAN) が推奨されるようになりました。

リリース 7.2 とリリース 7.3 の相違点

初期化パラメータ

次の初期化パラメータが Parallel Server Option に追加されました。

- CLEANUP_ROLLBACK_ENTRIES
- DELAYED_LOGGING_BLOCK_CLEANOUTS
- GC_FREELIST_GROUPS
- GC_RELEASABLE_LOCKS

データ・ディクショナリ・ビュー

次のビューが Parallel Server Option に追加されました。

- FILE_LOCK

動的パフォーマンス・ビュー

次のビューが変更されました。

- V\$BH

次のビューが追加されました。

- V\$SORT_SEGMENT
- V\$ACTIVE_INSTANCES

空きリスト・グループ

表およびクラスタと同様に、索引にも空きリスト・グループを設定できるようになりました。

ファイングレイイン・ロック

Oracle Parallel Server リリース 7.3 では、PCM ロックに、ファイングレイイン・ロックを使用して構成するためのオプションが追加されました。この変更によって、分散 Parallel Server キャッシュにあるデータベース・ブロックを保護するために使用されるロックを決定する様々なパラメータの解釈が影響を受けます。

ファイングレイイン・ロックは、マルチノード構成でのロックを提供する、より効果的な方法です。これは、特に MPP システムでのロック衝突の割合およびロック管理のための領域要件を削減します。この機能は、ハードウェアおよびオペレーティング・システム・プラットフォームが提供する機能に依存し、すべてのプラットフォームで使用可能ではありません。

インスタンス登録

この機能によって、各インスタンスは、インスタンス自身およびその属性のいくつかを登録でき、その他のインスタンスとの接続を確立できます。インスタンス登録は、Parallel Server のリモート・インスタンス上でパラレル実行に障害が発生した場合を除いて、ユーザーに対して透過的です。パラレル問合せがリモート・インスタンス上のエラーのために終了した場合は、障害インスタンスはエラー・メッセージで識別できるようになりました。

ソートの改善

このリリースでは、ソート一時領域の割当てが、より効率的な方法で行われるため、シリアル化およびインスタンス間の ping が削減されます。この機能を正しく設定すると、特にパラレル・モードでの Oracle Parallel Server のパフォーマンスが向上します。

最適な結果を得るには、安定したソート領域を確立してください。また、ソート領域はインスタンスにキャッシュされることに注意してください。インスタンスは、別のインスタンスに領域が不足し、最初のインスタンスに領域を解放するコールを発行するまで、領域を解放しません。これはコストがかかるシリアル化された処理であり、パフォーマンスを低下させることとなります。ご使用のシステムが安定したソート領域から常に逸脱する場合は、領域を多めに割り当てるか、または一時表領域を使用しないようにします。

ご使用のソート領域の安定度を判断するには、V\$SORT_SEGMENT ビューをチェックします。この新しいビューは、すべてのインスタンスのソート履歴を示します。FREED_EXTENTS 列および ADDED_EXTENTS 列が、割当て / 割当て解除アクティビティが過剰であることを示している場合は、対応する表領域に領域を追加することを考慮してください。また、ソート領域全体でインスタンス間の競合があるかどうかを判断するには、FREE_REQUESTS 値をチェックします。

過剰な割当ておよび割当て解除の別の原因として、ソート規模が大きすぎるものが考えられます。大規模なソートを必要とする操作に対しては、異なる一時表領域を割り当てると有効な場合があります。MAX_SORT_SIZE 値を参照すると、大規模なソートが実際に発生したかどうかを判断できる場合があります。

参照： ソート機能の拡張の詳細は、『Oracle8i パフォーマンスのための設計およびチューニング』を参照してください。

XA パフォーマンスの向上

スケーラビリティおよびスループットにおける様々な改善は、XA トランザクションに影響を及ぼします。これらの変更を行っても、パフォーマンスが向上する以外に、目立った影響はありません。

次の3つのラッチのパフォーマンスの大幅な向上によって、スケーラビリティが拡張しています。

- グローバル・トランザクション・マッピング表ラッチ
- エンキュー・ラッチ
- セッション切替えラッチ

ほとんどの共通 XA コールでコード・パスが削減され、データベースへの往復が削減されるため、トランザクション・スループットが拡張されます。

XA リカバリ拡張

Oracle Parallel Server では、XA インタフェースを使用して TP モニターを介して送信される分散トランザクションのリカバリは、完全にサポートされています。

XA_RECOVER コールが拡張され、別のインスタンスの障害トランザクションから、あるインスタンスの正しく完全なリカバリを確実にします。

XA_RECOVER コールをインスタンス・リカバリまで待機させるオプションが追加されています。この機能によって、障害が発生した Oracle インスタンスにかわって、別の Oracle インスタンスがリカバリを行うことができます（両方のインスタンスが同じ Oracle Parallel Server クラスタの一部である場合）。

XA_INFO 文字列には、OPS_FAILOVER という新しい句があります。この句が、特定の XA リソース・マネージャ接続に対して TRUE に設定されている場合、その接続から発行されたすべての XA_RECOVER コールは、必要なインスタンス・リカバリが完了するまで待機します。構文は次のとおりです。

OPS_FAILOVER=T

大文字または小文字（T または t）を使用できます。OPS_FAILOVER のデフォルト値は、FALSE（F または f）です。

以前は、XA_RECOVER コールが、障害インスタンスからインダウト・トランザクションのリストを戻すことは保証されていませんでした。OPS_FAILOVER=T と設定すると、必ずリストが戻るようになります。

OPS_FAILOVER が TRUE に設定されている場合、XA_RECOVER コールは、SMON がキャッシュ・リカバリを終了し、インダウト・トランザクションを識別し、そのトランザクションを、インダウト・トランザクションのリストがある PENDING_TRANS\$ 表に追加するまで待機します。

遅延トランザクション・リカバリ

トランザクション・リカバリの動作が変更され、次のことが可能になりました。

- 起動中のデータベース可用性の向上
- トランザクションの平行でのリカバリ（必要な場合）
- 短いトランザクションのリカバ리를妨げない、長いトランザクションのリカバリ

高速ウォームスタート

以前のリリースでは、障害発生後に完全にトランザクション・リカバリが実行されるまで、データベースをオープンできませんでした。リリース 7.3 では、キャッシュ・リカバリが完了するとすぐにデータベースは接続に対してオープンできます。（Oracle Parallel Server 環境でのフェイルオーバーとは異なり、データベースをオープンする場合のみ適用されます。）インスタンスに障害がある場合は、実行している別のインスタンスを介してデータベースに接続できます。

これは、障害が発生した時点でアクティブなトランザクションは、ロールバックされていないことを意味します。これらのトランザクションは、システムの利用者に対してアクティブである（行ロックを保持している）ように示されます。さらに、障害が発生した時点でアクティブなシステム全体のすべてのトランザクションは DEAD とマークされ、これらのトランザクションを含むロールバック・セグメントは PARTIALLY AVAILABLE とマークされます。これらのトランザクションは、SMON リカバリの一部としてバックグラウンドでリカバリされるか、またはこれらのトランザクションを検出したフォアグラウンド・プロセスによってリカバリされます（次の項で説明します）。ロールバック・セグメントは、オンラインで使用可能です。

トランザクション・リカバリ

高速ウォームスタート機能があると、すべてのトランザクションのリカバリに必要な時間によって、データベースの一般的な可用性が制限されません。リカバリされていないトランザクションによってロックされている部分を除いて、ユーザーはすべてのデータを使用できます。ただし、OLTP 作業負荷があると、データベースまたはインスタンスが停止したときにアクティブなすべての要求は、すぐに再送信されます。多くの場合、これらの要求は、リカバリされていないトランザクションが保持しているロックを検出します。そのため、これらのトランザクションのリカバリに必要な時間は、ロックされたデータにアクセスするために依然重要です。この問題を軽減するために、必要に応じてトランザクションを平行でリカバリすることができます。リカバリは次の操作によって行うことができます。

フォアグラウンド・プロセスによるリカバリ。まだリカバリされていないトランザクションによって行がロックされている場合があります。このような行を検出したフォアグラウンド・プロセスは、自力でトランザクションをリカバリできます。SMON によって行われる現在のリカバリも行われるため、これで全体的なトランザクション・リカバリが完了します。ただし、フォアグラウンド・プロセスが行ロックを検出すると、このプロセスはロックを保持しているトランザクションをすぐにリカバリし、継続できます。このようにして、リカバリ操作は必要に応じて平行化されます。停止したトランザクションは、SMON が

アクティブなトランザクションを妨げることはありません。以前は、アクティブなトランザクションは、停止したトランザクションをリカバリするまで、待機する必要がありました。

リカバリは、ロールバック・セグメントごとに行われます。このことによって、異なるインスタンスの複数のフォアグラウンド・プロセスが、同じロールバック・セグメントのトランザクションをリカバリする (ping を発生する) のを防ぎます。フォアグラウンド・プロセスはトランザクションを完全にリカバリします。リカバリされない場合は、そのトランザクションを待機するしかありませんでした。さらに、ロールバック・セグメント全体を検索して、リカバリされていないすべてのトランザクションを部分的にリカバリします。これは、構成可能な変更 (UNDO 記録) の数を、各トランザクションに適応します。このことによって、短いトランザクションは、長いトランザクションがリカバリされるのを待たずに、すぐにリカバリできます。初期化パラメータ CLEANUP_ROLLBACK_ENTRIES は、適用する変更の数を指定します。

SMON によるリカバリ。 SMON のトランザクション・リカバリ操作はほとんど変更されていません。SMON は、そのインスタンス内で DEAD とマークされているトランザクション、起動中のトランザクションおよびインスタンスをリカバリします。唯一の変更は、リカバリが必要なすべてのトランザクションを何度か検索し、その検索ごとに各トランザクションについて指定された UNDO 記録の数だけ適用することです。このことによって、短いトランザクションは、長いトランザクションのリカバ리를待機することがなくなります。

ロールバック・セグメントのオンライン化によるリカバリ。 ロールバック・セグメントのオンライン化によって、そのセグメントが含むすべてのトランザクションのリカバリが完全に行われます。以前は、プロセスのオンライン化によって、リカバリを行うよう SMON に通知していました。ウォームスタートまたはインスタンス起動の一部としてロールバック・セグメントを暗黙的にオンライン化すると、すべてのトランザクションをリカバリすることにはならず、かわりにそれらに DEAD とマークされることに注意してください。

接続時のロード・バランス

標準の Oracle では、ロード・バランスによって、複数のリスナーおよび複数のインスタンスが SQL*Net 接続時に均衡化されます。複数のリスナーが 1 つの Oracle インスタンスをリスニングでき、Oracle ディスパッチャが複数のリスナーに登録されます。SQL*Net クライアント・レイヤーは、DESCRIPTION_LIST 機能を介して複数のリスナーをランダム化します。

接続時のロード・バランスの詳細は、Oracle7 Server リリース 7.3 の SQL*Net ドキュメントを参照してください。

ソート・オペレーションのキャッシュの回避

`SORT_DIRECT_WRITES` 初期化パラメータのデフォルト値は AUTO です。これは、ソート領域が一定のサイズ以上の場合にオンになります。このことによって、パフォーマンスが向上します。詳細は、『Oracle8i パフォーマンスのための設計およびチューニング』を参照してください。

遅延ロギング・ブロック・クリーンアウト

Oracle7 Server リリース 7.3 では、遅延ブロック・クリーンアウトのパフォーマンスが向上し、関連する ping が削減されています。これらの拡張は、特に Oracle Parallel Server に対して効果があります。

Oracle7 Server リリース 7.3 では、新しい初期化パラメータ DELAYED_LOGGING_BLOCK_CLEANOUTS (デフォルトは TRUE) が提供されています。

Oracle がトランザクションをコミットすると、そのトランザクションが変更した各ブロックは、そのコミット時にはマークされません。各ブロックは、必要に応じて (ブロックが読み込まれまたは更新されたとき) 後でマークされます。このことを「ブロック・クリーンアウト」といいます。カレント・ブロックへの更新中にブロック・クリーンアウトが行われると、そのクリーンアウトは変更され、更新の REDO レコードには、更新の REDO レコードが付加されます。以前のリリースでは、カレント・ブロックへの読み込み中にブロック・クリーンアウトが必要な場合に、余分なクリーンアウト REDO レコードが生成され、ブロックが使用済になりました。これは変更されました。

リリース 7.3 では、トランザクションをコミットすると、そのトランザクションによって変更されたキャッシュのすべてのブロックがすぐにクリーンアウトされます。コミット時に実行されるこのクリーンアウトは高速バージョンであり、REDO ログ・レコードを生成せず、ブロックを再確保しません。ほとんどのブロックはこの方法でクリーンアウトされますが、長時間実行しているトランザクションが変更したブロックは除きます。

そのため、問合せ中は、データ・ブロックのトランザクション情報は通常最新の情報であり、ブロック・クリーンアウトが必要とされる頻度は大幅に減少します。通常ブロック・クリーンアウトは、トランザクションが実際にアクティブであるブロックを問い合わせる場合、またはコミット中にクリーンアウトされなかったブロックを問い合わせる場合に必要です。

変更時 (INSERT、DELETE、UPDATE) には、クリーンアウト REDO ログ・レコードが生成され、そのレコードには変更の REDO が付加されます。

パラレル実行プロセッサの親和性

Oracle7 Server リリース 7.3 では、サーバーがインスタンス間で、パラレル実行オプションに対して割り当てられるデフォルトの方法が改善されています。そのため、ユーザーは、ヒントを与えなくてもパラレル化を指定できます。

パラレル実行のスレーブは、ユーザー問合せに対するディスク転送率および CPU の処理率に基づいて割り当てられるようになっています。作業は、コストの高いリモート・ディスクよりも、ローカル・ディスクへアクセスするスレーブを問い合わせるように割り当てられます。この方法で、データのローカル性によって、パラレル実行のパフォーマンスが向上します。

最適な結果を得るには、データを、Parallel Server のインスタンスおよびノード間で均一に分割する必要があります。これは、特に、処理を大きく左右する中規模から大規模の表に関して必要です。データは、様々なディスク上またはすべてのノードに均一分散される必要があります。非常に小さな表では、必要ありません。

たとえば、2つのノードがある場合、1つのノードに 90%、もう1つのノードに 10% を常駐させるなど、均一でない方法で表を分割しないでください。同様に、4つのディスクがある場合、1つのディスクにデータの 90%、残りのディスクにデータの 10% が含まれるように分割しないでください。データは、使用可能なノードおよびディスク上に均一分散する必要があります。均一な分散は、ディスクのストライプ化を使用している場合は、自動的に行われます。ディスクのストライプ化を使用していない場合、パフォーマンスを最適化するには、手動でストライプ化を行う必要があります。

リリース 7.1 とリリース 7.2 の相違点

領域の事前割当ては不要

ほとんどの Parallel Server 構成では、空きリスト・グループ間のデータのパーティション化を維持するために、データ・ブロックを事前割当てする必要がなくなりました。行が挿入されると、データ・ブロックのグループは、インスタンスの適切な空きリスト・グループに割り当てられます。

データ・ディクショナリ・ビュー

次のビューが Parallel Server Option に追加されました。

- FILE_LOCK
- FILE_PING

動的パフォーマンス・ビュー

次のビューが変更されました。

- V\$BH
- V\$CACHE
- V\$PING
- V\$LOCK_ACTIVITY

次のビューが追加されました。

- V\$FALSE_PING
- V\$LOCKS_WITH_COLLISIONS
- V\$LOCK_ELEMENT

空きリスト・グループ

次のコマンドを使用して、特定のインスタンス、つまり空きリスト・グループをセッションから指定できるようになりました。

```
ALTER SESSION SET INSTANCE = instance_number
```

表ロック

次のコマンドを使用して、ユーザーが表をロックする機能を使用不可にできるようになりました。

```
ALTER TABLE table_name DISABLE TABLE LOCK
```

次のコマンドを使用して、表ロックを再度使用可能にできます。

```
ALTER TABLE table_name ENABLE TABLE LOCK
```

ロック・プロセス

障害が発生しているインスタンスが保持している PCM ロックが、そのインスタンスをリカバリするインスタンスのロック・プロセスでリカバリできるようになりました。

リリース7.0とリリース7.1の相違点

初期化パラメータ

- CACHE_SIZE_THRESHOLD が追加されました。

動的パフォーマンス・ビュー

次のビューが変更されました。

- V\$BH
- V\$CACHE
- V\$PING
- V\$LOCK_ACTIVITY

バージョン6とリリース7.0の相違点

この項では、Oracle バージョン6とOracle7リリース7.0の相違点について説明します。

バージョン互換性

バージョン6のParallel Server Optionは、Oracle7と上位互換性がありますが、これには例外が1つあります。バージョン6では、すべてのインスタンスが同じREDOログ・ファイルの集合を共有しますが、Oracle7では、各インスタンスに独自のREDOログ・ファイルの集合があります。Oracle7への移行の詳細は、『Oracle8i移行ガイド』を参照してください。Oracle7で動作できるようにデータベースがアップグレードされると、そのデータベースは、Oracleバージョン6のサーバーでは起動できません。Oracle7で実行できるアプリケーションが、Oracleバージョン6では実行できない場合があります。

ファイル操作

次のファイル操作は、Oracleバージョン6ではデータベースが排他モードでマウントされている場合しかサポートされていませんでしたが、Oracle7では、パラレル・モードでもサポートされています。

- データ・ファイルの追加、改名または削除
- データ・ファイルのオフライン化またはオンライン化
- 表領域の作成、変更または削除
- 表領域のオフライン化またはオンライン化

これらの操作を実行するインスタンスは、データベースをマウントしているだけでなく、オープンしている場合があります。

表 A-1 に、これらのファイル操作およびそれに対応する SQL 文を示します。Oracle バージョン 6 では、データベースをパラレル・モードでマウントしている場合は、これらの操作を行うことができません。

表 A-1 Oracle7 でサポートされるようになった SQL 文

操作	SQL 文
表領域の作成	CREATE TABLESPACE tablespace_name
表領域の削除	DROP TABLESPACE tablespace_name
表領域のオフライン化またはオンライン化	ALTER TABLESPACE tablespace OFFLINE ALTER TABLESPACE tablespace ONLINE
データ・ファイルの追加	ALTER TABLESPACE tablespace ADD DATAFILE
データ・ファイルの改名	ALTER TABLESPACE tablespace RENAME DATAFILE
データ・ファイルのログ・ファイルの改名	ALTER TABLESPACE tablespace RENAME FILE
REDO ログ・ファイルの追加	ALTER DATABASE dbname ADD LOGFILE
REDO ログ・ファイルの削除	ALTER DATABASE dbname DROP LOGFILE
データ・ファイルのオフライン化またはオンライン化	ALTER DATABASE dbname DATAFILE OFFLINE ALTER DATABASE dbname DATAFILE ONLINE

Oracle7 では、データベースが共有モードでマウントされていても、前述したすべてのファイル操作が実行できます。

REDO ログ・ファイルは、アクティブであるとき、または、削除するとスレッドのグループ数が 2 より少なくなる場合は、削除できません。データ・ファイルを Oracle7 でオンラインまたはオフラインにすると、インスタンスはデータベースをオープンまたはクローズして、マウントできます。他のインスタンスがデータベースをオープンしている場合は、ファイルをオンラインまたはオフラインにするインスタンスも、データベースをオープンしている必要があります。

注意： データ・ファイルの追加、表領域の作成、または表領域およびそのデータ・ファイルの削除を行った場合は、Oracle をパラレル・モードで再起動する前に、必要に応じて GC_FILES_TO_LOCKS および GC_DB_LOCKS の値を調整する必要があります。調整しない場合、新しいファイルを処理するロックの数が不足する場合があります。

遅延ロールバック・セグメント

グローバル定数パラメータ `GC_SAVE_ROLLBACK_LOCKS` は、遅延ロールバック・セグメント用の分散ロックを確保します。この遅延ロールバック・セグメントには、オフラインにされた表領域のトランザクションに対するロールバック・エントリが含まれます。

バージョン6では、パラレル・モードでの表領域のオフライン化はサポートされていません。そのため、初期化パラメータ `GC_SAVE_ROLLBACK_LOCKS` は Oracle バージョン6では必要ありません。Oracle7では、このパラメータは遅延ロールバック・セグメント用に必要です。

REDO ログ

Oracle バージョン6では、すべてのインスタンスが同じ REDO ログ・ファイルの集合を共有し、各インスタンスは、カレント REDO ログ・ファイル内でそれに割り当てられた領域に書き込みます。

Oracle7では、各インスタンスに独自の REDO ログ・ファイルの集合があります。REDO ログ・ファイルの集合を「REDO スレッド」といいます。スレッド番号は、REDO ログ・ファイルがデータベースに追加されるときにそのファイルに対応付けられます。各インスタンスは、起動時にスレッド番号を取得します。

Oracle7では、ログ・スイッチはインスタンスごとに実行されます。Oracle バージョン6では、インスタンスが REDO ログ・ファイルを共有するため、ログ・スイッチはすべてのインスタンスに適用されます。

Oracle7では、オンライン REDO ログ・ファイルのミラー化が導入されています。ミラー化の程度は、インスタンスごとに決定されます。これによって、各インスタンスで実行するアプリケーションの要件に従って、ミラー化を指定できます。

ALTER SYSTEM SWITCH LOGFILE

Oracle バージョン6では、すべてのインスタンスが、1つのオンライン REDO ログ・ファイルの集合を共有していました。そのため、ALTER SYSTEM SWITCH LOGFILE 文によって、すべてのインスタンスで新しい REDO ログ・ファイルへのログ・スイッチが強制的に行われました。

Oracle7では、この SQL 文に対するグローバル・オプションはありませんが、ALTER SYSTEM ARCHIVE LOG CURRENT 文を使用して、すべてのインスタンスでログ・ファイルを強制的に切り替える（および切替えが行われるまでのすべてのオンライン・ログ・ファイルをアーカイブする）ことができます。

初期化パラメータ

Oracle バージョン6の `LOG_ALLOCATION` パラメータは、Oracle7では廃止されました。Oracle7には、新しい初期化パラメータ `THREAD` が含まれています。このパラメータは、起動時に REDO ログ・ファイルの集合を特定のインスタンスに対応付けます。

空き領域リスト

この項では、空き領域リストに関する変更について説明します。

削除および更新によって解放された領域

Oracle バージョン6では、削除または行を縮小する更新によって解放されたブロックは、空き領域の共通プールに追加されます。Oracle7では、ブロックは、そのブロックを削除したプロセスの空きリストおよび空きリスト・グループに追加されます。

クラスタの空きリスト

Oracle バージョン6では、FREELISTS および FREELIST GROUPS 記憶域オプションは、CREATE CLUSTER 文で使用できませんでした。また、ALLOCATE EXTENT 句は、ALTER CLUSTER 文で使用できませんでした。

Oracle7では、CREATE CLUSTER の FREELISTS および FREELIST GROUPS 記憶域オプションを指定し、ALTER CLUSTER ALLOCATE EXTENT (INSTANCE *n*) 文でエクステンツをインスタンスに割り当てることによって、クラスタ（大部分のハッシュ・クラスタは除く）が複数の空きリストを使用できます。

Oracle7では、ハッシュ・クラスタがハッシュ関数に対するユーザー定義キーで作成され、そのキーがインスタンスによってパーティション化されている場合、ハッシュ・クラスタは空きリストおよび空きリスト・グループを持つことができます。

初期化パラメータ

FREELISTS および FREELIST GROUPS 記憶域オプションが、Oracle バージョン6の初期化パラメータ FREE_LIST_INST および FREE_LIST_PROC のかわりに使用されます。

インポート/エクスポート

Oracle バージョン6では、エクスポートは空きリスト情報をエクスポートしませんでした。Oracle7では、エクスポートおよびインポートは FREELISTS および FREELIST GROUPS を処理できます。

SQL*DBA

バージョン6では、切断中に STARTUP および SHUTDOWN を実行する必要がありました。Oracle7 リリース7.0では、INTERNAL、SYSDBA または SYSOPER として接続中に、STARTUP および SHUTDOWN を発行する必要があります。

Oracle7では、操作は、コマンドまたは SQL*DBA メニュー・インタフェースのいずれかを使用して実行することができます。詳細は、『Oracle8i ユーティリティ・ガイド』を参照してください。

初期化パラメータ

この項では、新しいパラメータおよび廃止されたパラメータについて説明します。

新しいパラメータ

新しい初期化パラメータ `THREAD` は、起動時に REDO ログ・ファイルの集合を特定のインスタンスに対応付けます。

新しいパラメータの完全なリストは、『Oracle8i リファレンス・マニュアル』を参照してください。

廃止されたパラメータ

以前のバージョンの Parallel Server Option で使用されていた次の初期化パラメータは、Oracle7 では廃止されました。

- `ENQUEUE_DEBUG_MULTI_INSTANCE`
- `FREE_LIST_INST`
- `FREE_LIST_PROC`
- `GC_SORT_LOCKS`
- `INSTANCES`
- `LANGUAGE`
- `LOG_ALLOCATION`
- `LOG_DEBUG_MULTI_INSTANCE`
- `MI_BG_PROCS` (`GC_LCK_PROCS` へ改名)
- `ROW_CACHE_ENQUEUE`
- `ROW_CACHE_MULTI_INSTANCE`

廃止されたパラメータの完全なリストは、『Oracle8i 移行ガイド』を参照してください。

アーカイブ

Oracle バージョン6 では、すべてのインスタンスが同じ REDO ログ・ファイルを共有するため、各インスタンスは Parallel Server 全体のオンライン REDO ログ・ファイルをアーカイブします。そのため、記憶域メディアへのアクセスが最も容易なインスタンスに自動アーカイブを使用させ、その他のインスタンスには手動でアーカイブさせることが可能です。

Oracle7 では、各インスタンスに独自の REDO ログ・ファイルの集合があるため、自動アーカイブはカレント・インスタンスに対してのみ行われます。Oracle7 は、クローズされているスレッドもアーカイブできます。手動アーカイブによって、すべてのインスタンスのオンライン REDO ログ・ファイルをアーカイブできます。ALTER SYSTEM ARCHIVE LOG 文の

THREAD オプションを使用すると、任意の特定のインスタンスの REDO ログ・ファイルをアーカイブできます。

Oracle7では、アーカイブ REDO ログ・ファイルのファイル名に、スレッド番号およびログ順序番号を含めることができます。

新しい初期化パラメータ LOG_ARCHIVE_FORMAT は、アーカイブされたファイル名のフォーマットを指定します。CREATE DATABASE 文で使用される新しいデータベース・パラメータ MAXLOGHISTORY によって、制御ファイルにアーカイブ履歴を保持するように指定できます。

メディア・リカバリ

Oracle7では、メディア障害からのオンライン・リカバリは、データベースがパラレル・モードまたは排他モードのいずれでマウントされている場合でもサポートされます。

いずれのモードでも、リカバリされているデータベースまたはオブジェクトは、リカバリ中は使用できません。

- データベース全体をリカバリするには、データベースはマウントされている必要がありますが、オープンされている必要はありません。
- 表領域をリカバリするには、データベースはオープンされ、表領域はオフラインである必要があります。
- データ・ファイル (SYSTEM 表領域のファイル以外) をリカバリするには、データ・ファイルがオフライン状態で、データベースがクローズまたはオープンされている必要があります。

B

制限事項

この付録では、Oracle Parallel Server の互換性の問題および制限事項について説明します。
この付録の内容は次のとおりです。

- 共有モードと排他モード間の互換性
- 制限事項

共有モードと排他モード間の互換性

次の項では、Parallel Server での共有モードと排他モード間の互換性について説明します。

- [エクスポート・ユーティリティおよびインポート・ユーティリティ](#)
- [共有モードと排他モード間の互換性](#)

エクスポート・ユーティリティおよびインポート・ユーティリティ

エクスポート・ユーティリティは Oracle データベースからオペレーティング・システム・ファイルヘデータを書き込み、インポート・ユーティリティはこれらのオペレーティング・システム・ファイルから Oracle データベースヘデータを読み込みます。Oracle のこの機能は、共有モードでも排他モードでも同じです。

参照： インポートおよびエクスポートの詳細は、『Oracle8i ユーティリティ・ガイド』を参照してください。

共有モードと排他モード間の互換性

Oracle Parallel Server は、排他モードで作成されたすべての Oracle データベースで動作します。各インスタンスには、インスタンス自身の REDO ログの集合が必要です。

排他モードの Oracle は、Oracle Parallel Server が作成または変更したデータベースにアクセスできます。

Oracle Parallel Server が空き領域を特定のインスタンスに割り当てると、その領域は、排他モードの別のインスタンスの挿入には使用できない場合があります。割り当てられたエクステンツのすべてのデータは、常に使用可能です。

制限事項

次の項では、制限事項について説明します。

- [一度に割り当てられるブロックの最大数](#)
- [共有モードでの制限事項](#)

一度に割り当てられるブロックの最大数

GC_FILES_TO_LOCKS パラメータの *!blocks* オプションによって、空きリスト・グループ内で使用できるブロック数を制御できます。*!blocks* を使用して、エクステンツ内に割り当てられるブロックの割合（一度に 255 ブロックまで）を指定できます。

共有モードでの制限事項

共有モードで複数のインスタンスを実行している Oracle は、排他モードでの Oracle のすべての機能をサポートします。ただし、次に説明するものを除きます。

制限付き SQL 文

共有モードでは、次の操作がサポートされません。

- データベースの作成 (CREATE DATABASE)
- 制御ファイルの作成 (CREATE CONTROLFILE)
- データベースのアーカイブ・モードの切替え (ALTER DATABASE の ARCHIVELOG および NOARCHIVELOG オプション)

これらの操作を実行するには、すべてのインスタンスを停止し、排他モードで1つのインスタンスを起動します。

データ・ファイルの最大数

Oracle がサポートするデータ・ファイルの数は、オペレーティング・システムによって異なります。この制限内では、許可される最大値は CREATE DATABASE コマンドで使用される値によって異なり、このコマンドは制御ファイルの物理サイズによって制限されます。この制限は排他モードと共有モードで同じですが、Oracle Parallel Server の追加インスタンスは、単一インスタンスのシステムよりも、ファイルの最大数を制限します。詳細は、『Oracle8i SQL リファレンス』およびオペレーティング・システム固有の Oracle マニュアルを参照してください。

シーケンス・ジェネレータ

Oracle Parallel Server は、共有モードでのシーケンス・ジェネレータの CACHE オプションと ORDER オプションの組合せをサポートしていません。これらのオプションを両方使用して作成された順序は順序付けされますが、Parallel Server で実行中の場合はキャッシュされません。

インポート・ユーティリティおよびエクスポート・ユーティリティでの空きリスト

エクスポート・ユーティリティは、複数の空きリストおよび空きリスト・グループに関する情報を保持しません。複数のインスタンスからデータをエクスポートし、単一ノードからそのデータをファイルにインポートすると、そのデータは、最初と同じ状態でエクステントに分散されない場合があります。データがインポートされた表のメタデータは、データ・ブロックに対応付けられた空きリストおよび空きリスト・グループの情報を含んでいます。

そのため、エクスポートおよびインポートを使用してデータをバックアップおよびリストアする場合は、データが再度パーティション化されるようにそのデータをインポートするのは困難です。

索引

数字

- 1 対 1 ロック, 5-15, 5-18, 5-20, 5-32
 - 解放可能, 5-5
 - 固定, 5-5
 - 使用する場合, 5-32
- 1 対 n ロック, 5-5
 - 解放可能, 5-5
 - 固定, 5-5
- 9 の数
 - 可用性を評価する基準, 9-3

A

- ACTIVE_INSTANCE_COUNT, 9-18
- ADD LOGFILE 句
 - THREAD 句, 7-3
- ADDED_EXTENTS, A-12
- ALERT ファイル, 7-2
- ALLOCATE EXTENT オプション
 - DATAFILE オプション, 7-24
 - INSTANCE オプション, 7-24
 - SIZE オプション, 7-24
 - インスタンス番号, 7-20
 - 使用不可能, A-22
- ALLOW_PARTIAL_SN_RESULTS パラメータ
 - リリース 8.1 で廃止, A-6
- ALTER CLUSTER 文, A-22
- ALTER DATABASE 文
 - ADD LOGFILE, 7-3
 - DATAFILE OFFLINE および ONLINE オプション, A-20
 - ファイルの改名, A-20
 - ログ・ファイルの追加または削除, A-20
 - ログ・モードの設定, B-3

- ALTER ROLLBACK SEGMENT コマンド, 7-8
- ALTER SESSION 文
 - SET INSTANCE オプション, 7-24
- ALTER SYSTEM ARCHIVE LOG 文
 - CURRENT オプション, A-21
- ALTER SYSTEM CHECK DATAFILES 文, 7-2
- ALTER SYSTEM SWITCH LOGFILE 文, A-21
 - DBA 権限, A-21
- ALTER TABLESPACE 文
 - ADD DATAFILE オプション, A-20
 - OFFLINE および ONLINE オプション, A-20
 - データ・ファイルの改名, A-20
- ALTER TABLE 文
 - DISABLE TABLE LOCK オプション, 4-13
 - ENABLE TABLE LOCK オプション, 4-13
- ARCHIVE LOG 句
 - CURRENT オプション, A-21
- ARCHIVELOG モード
 - オンライン・バックアップおよびオフライン・バックアップ, 1-5
 - 自動アーカイブ, 1-5
 - モードの変更, B-3
- AST
 - 定義, 5-27

B

- BSP プロセス, 6-2
 - 定義, 6-5

C

- CACHE_SIZE_THRESHOLD パラメータ, A-19
 - リリース 8.1 で廃止, A-6
- CACHE オプション、CREATE SEQUENCE, 8-14

CHECK DATAFILES 句, 7-2
CLEANUP_ROLLBACK_ENTRIES パラメータ, A-11,
A-15
Cluster Manager, 9-6
Node Monitor, 3-3
目的, 3-3
Cluster Manager ソフトウェア
目的, 3-3
CM
DLM との相互作用, 3-6
Connect-Time Failover, 8-10
CPU
単一、ユニプロセッサ, 2-2
追加, 2-3
CREATE CLUSTER 文, A-22
CREATE CONTROLFILE 文
排他モード, B-3
CREATE DATABASE 文
MAXINSTANCES, 7-21
MAXLOGHISTORY, 7-5
排他モード, B-3
CREATE SEQUENCE 文, 8-14
CACHE オプション, 8-14
CYCLE オプション, 8-14
ORDER オプション, 8-14
説明, 8-13
CREATE TABLESPACE 文, A-20
CREATE TABLE 文
FREELIST GROUPS オプション, 7-24
FREELISTS オプション, 7-24
CR サーバー
リリース 8.1 の変更概要, A-3
CURRENT オプション
Oracle7 で新規, A-21
CYCLE オプション、CREATE SEQUENCE, 8-14

D

Database Configuration Assistant
ログ・ファイル, 7-5
DATAFILE オプション
表領域, A-20
DB_BLOCK_BUFFERS パラメータ, 5-32
DBA_ROLLBACK_SEGS ビュー, 7-7
DELAYED_LOGGING_BLOCK_CLEANOUTS パラ
メータ, A-11, A-16
DESCRIPTION_LIST 機能, A-15

DLM, 5-1
CM との相互作用, 3-6
DML_LOCKS パラメータ, 4-13
DML ロック, 4-4, 4-5
DM、データベース・マウント, 4-8
DROP TABLESPACE 文, A-20
DROP TABLE 文, 4-8
DSS アプリケーション, 8-6, 8-7

E

ENQUEUE_DEBUG_MULTI_INSTANCE パラメータ
(Oracle バージョン 6), A-23
EXCLUSIVE パラメータ
リリース 8.0.4 で廃止, A-7

F

false ping, 5-31
FAST_START_PARALLEL_ROLLBACK, A-4
FDDI
Oracle Parallel Server で使用する場合, 2-5
Fibre Channel
Oracle Parallel Server で使用する場合, 2-6
FILE_LOCK ビュー, A-11
FREE_LIST_INST パラメータ (Oracle バージョン 6),
A-22, A-23
FREE_LIST_PROC パラメータ (Oracle バージョン 6),
A-22, A-23
FREED_EXTENTS, A-12
FREELIST GROUPS 記憶域オプション
インスタンス番号, 7-20
クラスタ化表, A-22
FREELISTS
パラメータ, 7-12
FREELISTS 記憶域オプション
クラスタ化表, A-22
FREEZE_DB_FOR_FAST_INSTANCE_RECOVERY パ
ラメータ, A-8

G

GC_DB_LOCKS パラメータ, A-8
ファイル操作後の調整, A-20
GC_FILES_TO_LOCKS パラメータ, 5-20, 5-31
PCM ロックとエクステントの対応付け, 7-21
ファイル操作後の調整, A-20

GC_FREELIST_GROUPS パラメータ, A-8, A-11
GC_LATCHES パラメータ, A-6
リリース 8.1 で廃止, A-6
GC_LCK_PROCS パラメータ
リリース 8.1 で廃止, A-6
GC_RELEASABLE_LOCKS パラメータ, A-11
GC_ROLLBACK_LOCKS パラメータ, 7-7
GC_ROLLBACK_SEGMENTS パラメータ, A-8
分散ロック数, 7-7
GC_SAVE_ROLLBACK_LOCKS パラメータ, 7-7,
A-8, A-21
GC_SEGMENTS パラメータ, A-8
GC_SORT_LOCKS パラメータ, A-23
GC_TABLESPACES パラメータ, A-8
GLOBAL オプション
ファイルへのアクセスの検証, 7-2
GMS
リリース 8.1 で削除, A-4
GV\$ ビュー, A-9

I

INITIAL 記憶域パラメータ
ロールバック・セグメント, 7-7
INSTANCE_NUMBER パラメータ
SQL オプション, 7-24
インスタンスへの空きリストの割当て, 7-23
INSTANCES パラメータ (Oracle バージョン 6), A-23
I/O
最小化, 1-4, 4-9
ディスク競合, 1-4
割込み, 8-11

L

Lampport SCN 生成, 6-6
LANGUAGE パラメータ (Oracle バージョン 6), A-23
LCKn プロセス, 5-17
LM_LOCKS パラメータ, A-8
LM_PROCS パラメータ, A-8
LM_RESS パラメータ, A-8
LMDn プロセス, A-9
LMON
クラスタ再編成, 9-14
LMON プロセス, A-9
LOCAL オプション
ファイルへのアクセスの検証, 7-2

LOG_ALLOCATION パラメータ (Oracle バージョン
6), A-21, A-23
LOG_DEBUG_MULTI_INSTANCE パラメータ
(Oracle バージョン 6), A-23
LOG_FILES パラメータ
リリース 8.1 で廃止, A-6
LRU リスト
ロック要素, 5-19

M

MAX_COMMIT_PROPAGATION_DELAY パラメータ,
6-6
MAX_SORT_SIZE, A-12
MAXEXTENTS 記憶域パラメータ
自動割当て, 7-25
MAXINSTANCES オプション, 7-21
MAXINSTANCES パラメータ, 7-23
インスタンスへの空きリストの割当て, 7-20, 7-23
MAXLOGHISTORY オプション, 7-5
MINEXTENTS 記憶域パラメータ
自動割当て, 7-25
modulo, 7-20, 7-23
MTBF
定義, 9-2
MTS_LISTENER_ADDRESS パラメータ
リリース 8.0.4 で廃止, A-7
MTS_MULTIPLE_LISTENERS パラメータ
リリース 8.0.4 で廃止, A-7
MTTR, 9-5
定義, 9-2

N

NEXT 記憶域パラメータ, 7-7
NOARCHIVELOG モード
オフライン・バックアップ, 1-5
モードの変更, B-3
Node Monitor, 3-3
NOORDER オプション, CREATE SEQUENCE, 8-14
NULL ロック・モード, 4-12
NUMA
定義, 2-3
N ノード
Oracle Parallel Server 構成, 9-17

O

- OGMS_HOME パラメータ, A-7
 - リリース 8.1 で廃止, A-6
- OLTP, 8-15
- OLTP アプリケーション, 8-5, 8-6, 8-7, A-3
- OPS_ADMIN_GROUP パラメータ, A-8
 - リリース 8.1 で廃止, A-6
- OPS_FAILOVER 句, A-13
- Oracle
 - 移行, A-19
 - 互換性, B-3
 - 制限, 8-14
 - 制限事項, A-19, A-21
 - データ・ディクショナリ, 4-2
 - データ・ファイルの互換性, 7-2
 - 廃止されたパラメータ, A-23
 - バックアップ, 1-5
 - パフォーマンス機能, 1-4
- Oracle Parallel Server
 - Connect-Time Failover, 8-10
 - Oracle パラレル実行, 8-15
 - アプリケーション・システム・タイプ, 1-2
 - 概要, 1-1
 - 効果, 1-2
 - 定義, 1-2
- Oracle Parallel Server Management (OPSM), A-5
- Oracle Parallel Server フェイルオーバー
 - 動作方法, 9-13
- oracle_pid, 7-23
- Oracle パラレル実行
 - 説明, 8-15
- Oradebug, A-5
- ORDER オプション, 8-14

P

- Parallel Server, A-5
 - Oradebug, A-5
 - インストール, A-5
 - ジョブのインスタンス親和性, A-6
 - データベース構成, A-5
 - リスナー・ロード・バランス, A-5
- PARALLEL_DEFAULT_MAX_INSTANCES パラメータ
 - リリース 8.1 で廃止, A-6

- PARALLEL_SERVER パラメータ
 - リリース 8.0 で新規, A-7
- PARALLEL_TRANSACTION_RECOVERY パラメータ
 - リリース 8.1 で変更, A-4
- PARALLEL パラメータ
 - リリース 8.0.4 で廃止, A-7
- PCM
 - 1 対 1 ロック, 5-5
 - 1 対 1 ロックの使用法, 5-32
 - 1 対 n ロック, 5-5
 - ハッシュ・ロックの使用法, 5-32
- PCM ロック
 - 解放可能, 5-4
 - 競合, 7-21
 - 固定 1
 - n, 5-4
 - ファイル集合, 5-21
 - ブロックのマッピング, 5-20, 7-21
- PCTFREE, 7-12
- PCTUSED, 7-12
- ping, 5-22, 5-24
 - false, 5-31
 - 定義, 5-2

R

- RAID, 9-8
- REDO ログ・ファイル
 - REDO スレッド, 7-3
 - アーカイブ, 1-5
 - 上書き, 1-5
 - 改名, A-20
 - 削除, A-20
 - 制御ファイル内の識別, 7-5
 - 追加, A-20
- REDO ログ・ファイルのアーカイブ
 - オンライン・アーカイブ, 1-5
 - 制御ファイル内の識別, 7-5
- REDO ログ・ファイルの削除, A-20
- ROLLBACK_SEGMENTS パラメータ, 7-7, 7-8
- ROW_CACHE_MULTI_INSTANCE パラメータ
 - (Oracle バージョン 6), A-23

S

- SCN, 6-6

SCSI

- Oracle Parallel Server で使用する場合, 2-6
- SEQUENCE_CACHE_ENTRIES パラメータ
リリース 8.1 で廃止, A-6
- SMON プロセス, 6-3
トランザクション・リカバリ, A-15
- SMP
制限事項のスケール, 2-3
- SORT_DIRECT_WRITES パラメータ, A-15
- SQL 文
制限事項, B-3
- SYSTEM ロールバック・セグメント, 7-6

T

- THREAD オプション
パブリック・スレッドの作成, 7-3
プライベート・スレッドの作成, 7-3
- THREAD パラメータ
インスタンス獲得スレッド, 7-3
- TM、DML エンキュー, 4-7
- TP モニター, A-13
- TRANSACTIONS_PER_ROLLBACK パラメータ, 7-8
- TRANSACTIONS パラメータ, 7-8
- TX、トランザクション, 4-7

U

- UMA
定義, 2-3

V

- V\$ACTIVE_INSTANCES ビュー, A-11
- V\$BH ビュー, 5-24, A-9, A-11, A-19
- V\$CACHE ビュー, A-19
- V\$CLASS_PING ビュー, A-9
- V\$DATAFILE ビュー, 7-3
- V\$DLM_ALL_LOCKS
リリース 8.1 で新しいビュー, A-3
- V\$DLM_CONVERT_LOCAL ビュー, A-9
- V\$DLM_CONVERT_REMOTE ビュー, A-9
- V\$DLM_LATCH ビュー, A-9
- V\$DLM_LOCKS ビュー
リリース 8.0.4 で変更, A-7
- V\$DLM_MISC ビュー, A-9

V\$DLM_RESS

- 新しいビュー, A-3
- V\$FAST_START_SERVERS
ビュー, A-4
- V\$FAST_START_TRANSACTIONS
ビュー, A-4
- V\$FILE_PING ビュー, A-9
- V\$LOCK_ACTIVITY ビュー, A-19
- V\$LOCKS_WITH_COLLISIONS ビュー, A-18
- V\$LOGFILE ビュー, 7-5
- V\$PING ビュー, A-19
- V\$RESOURCE_LIMIT ビュー, A-9
- V\$SORT_SEGMENT ビュー, A-11, A-12
- V\$SYSSTAT ビュー, A-9

X

- XA_RECOVER コール, A-13
- XA インタフェース
パフォーマンス強化, A-13
ライブラリ, 5-33
リカバリ拡張, A-13

あ

- アーカイブ・モードの切替え, B-3
- アーキテクチャ
Oracle Parallel Server のコンポーネント, 3-1
パラレル処理, 2-1
- 空きリスト, 7-21
PCM ロック, 7-21
インスタンスへの割当て, 7-23
エクスポート・ユーティリティ, 7-19, B-3
クラスタ, A-22
排他モード, B-2
- 空きリスト・グループ
インスタンスへの割当て, 7-20, 7-23
定義, 7-13
リリース 7.3 で拡張, A-11
- アプリケーション
DSS, 8-6
OLTP, 8-6
拡張しない, 9-23
スケーラビリティ, 8-10, 8-12
挿入集中型, 7-19
チューニング・パフォーマンス, 7-19
問合せ処理集中型, 8-6

パーティション化データ, 8-6
パフォーマンス, 7-19
部門別, 8-7
プロファイル, 8-7
ランダム・アクセス, 8-6

い

イーサネット

Oracle Parallel Server で使用する場合, 2-5

移行

データ移行, B-2

一貫性

複数バージョン読み込み, 1-4

ロールバック情報, 7-6

インスタンス

PCM ロックのオーナーシップ, 5-17

最大数, 7-6, 7-21

親和性、ジョブ, A-6

インスタンス番号, 7-20

スレッド番号, 7-3

データ・ブロックとの対応付け, 7-24

必要なロールバック・セグメント, 7-6

リカバリ, 9-15

インスタンス登録, A-12

インスタンスの停止

順序番号の消失, 8-14

インスタンス番号, 7-23

インスタンス・リカバリ

ファイルへのアクセス, 7-3

ロールバック・セグメント, 7-6

インターコネクト

クラスタ・コンポーネントとして, 2-2

高速, 4-2

冗長性, 9-6

スケーラビリティ, 8-11

定義, 2-4

待ち時間, 2-7

インポート・ユーティリティ

空きリスト, A-22

互換性, B-2

データのリストア, B-2

え

エクステント

PCM ロックの割当て, 7-21

インスタンスに割り当てられない, 7-25
サイズ, 7-7

ロールバック・セグメント, 7-7

エクステントの事前割当て, 7-25

エクスポート・ユーティリティ

空きリスト, 7-19, A-22, B-3

互換性, B-2

データのバックアップ, B-2

エンキュー, 4-3

OPS コンテキスト, 4-5

グローバル, 4-4

ローカル, 4-4

お

オフライン・データ・ファイル, A-20

オフライン・バックアップ, 1-5

オフライン表領域

制限事項, 7-6, A-20

遅延ロールバック・セグメント, A-21

オペレーティング・システム

エクスポートされたファイル, B-2

スケーラビリティ, 8-11

オペレーティング・システム固有のレイヤー

プロセス間通信 (IPC), 6-5

オペレーティング・システム

分散ロック・マネージャ, 5-22

オンライン・REDO ログ・ファイル

REDO スレッド, 7-3

オンライン・アーカイブ, 1-5

オンライン・データ・ファイル

サポートされている操作, A-20

オンライン・バックアップ, 1-5

オンライン・リカバリ, A-24, 7-3

か

解放可能ロック

固定との比較, 5-5

作成, 5-4

概要

Oracle Parallel Server, 1-1

拡張

パーティション, 1-2

獲得 AST, 5-27, 5-30

可用性

インターコネクト, 2-5

共有ディスク・システム, 2-6
データ・ファイル, 7-3
パラレル・データベースの利点, 1-3
評価, 9-2
容量計画, 9-4

き

記憶域オプション

エクステンント・サイズ, 7-7
クラスタ化表, A-22
ロールバック・セグメント, 7-7

起動

共有モード, A-20
グローバル定数パラメータ, 7-7
ファイル操作後, A-20
ファイルへのアクセスの検証, 7-2
ロールバック・セグメント, 7-6

機能

新規, ix

機能、新規

プライマリ / セカンダリ・インスタンス, A-2

キャッシュ

一貫性, 4-9, 5-2
行キャッシュ, 4-2
順序キャッシュ, 8-13, 8-14
ディクショナリ・キャッシュ, 4-2
ディクショナリのフラッシュ, 4-8
バッファ・キャッシュ, 1-4

キャッシュ・フュージョン

リリース 8.1 の変更概要, A-3

キャッシュ・リカバリ, 9-16

キャッシュ・フュージョン

アーキテクチャ, 6-5
定義, 6-5

行キャッシュ, 4-2

競合

順序番号, 8-13
ディスク, 7-2, 7-7
表データ, 7-2, 7-6
ブロック, 7-7, 7-21
ロールバック・セグメント, 7-6, 7-7

共有ディスク・システム

メリット, 2-6

共有排他モード, 5-9

共有メモリー・アクセス

メリット、デメリット, 2-4

共有メモリー・システム

スケラビリティ, 8-11

共有モード

データ・ファイル, 7-2

ファイル操作の制限事項, A-20

行レベル・ロック

DML ロック, 4-7

PCM ロックの独立性, 4-12

システム共有リソース, 1-4

許可キュー, 5-28, 5-30

均一ディスク・アクセス, 2-5

均一メモリー・アクセス

定義, 2-3

<

クライアント

フェイルオーバー, 9-9

ランダム化, 8-10

ロード・バランス, 8-10

クライアント・ロード・バランス, 8-10

クラスタ

コンポーネント, 2-2

実装, 8-11

ストレージ・アクセス, 2-5

定義, 1-2

ハッシュ・クラスタ, A-22

パフォーマンス, 8-7

クラスタ化システムでのストレージ・アクセス, 2-5

クラスタ再編成, 9-14

グループ

REDO ログ・ファイル, 7-4

V\$LOGFILE, 7-5

一意の番号, 7-5

グループ・ベースのロック, 5-33

グループ・メンバーシップ・サービス, A-10

リリース 8.1 で削除, A-4

グローバル定数パラメータ

非 PCM ロック, 4-9

ロールバック・セグメント, 7-7

グローバル動的パフォーマンス・ビュー, A-9

グローバル・ロック, 4-4

モード, 4-12

クロス登録, 9-21

け

計画

- 冗長性, 9-4
- 容量, 9-4

こ

高可用性

- Cluster Manager, 3-3
- 構成, 9-17
- 定義, 1-3, 9-2
- パラレル・データベースの効果, 1-3

高可用性の共有ノード構成, 9-24

更新

- PCM ロック, 4-12
- インスタンス・ロック, 5-22
- 異なる時間での, 8-6
- 同時, 1-4
- パフォーマンス, 7-19

構成

- 高可用性, 9-17

互換性

- 共有および排他モード, 7-2

固定モード、ロック要素, 5-19

固定ロック

- 解放可能との比較, 5-5

コミットされたデータ

- 順序番号, 8-14

コンポーネント

- Oracle Parallel Server, 6-1

さ

サーバー側フェイルオーバー, 9-12

最高水位標, 7-25

- 移動, 7-26, 7-27
- 定義, 7-11, 7-26

再マスター化

- フェイルオーバー中, 9-8
- ロック, 9-15

作業負荷

- スケールアップ, 8-2

し

シーケンス・ジェネレータ

- LM ロック, 8-13
- Parallel Server における, 8-13
- アプリケーション・スケーラビリティ, 8-12
- 順序番号のスキップ, 8-14
- 制限, 8-14

制限事項, B-3

分散ロック, 8-13

シェアード・ナッシング・システム, 2-7

システム・グローバル領域 (SGA)

行キャッシュ, 4-2

順序キャッシュ, 8-14

システム固有の Oracle マニュアル

空きリスト・オーバーヘッド, 7-12

データ・ファイル、最大数, B-3

システム変更番号 (SCN)

Lampport, 6-6

増加, 6-6

非 PCM ロック, 4-5

システム・レベル計画

計画

- システム・レベル, 9-4

持続リソース, 3-6

順序

- キャッシュされない, B-3, 8-14

タイムスタンプ, 8-14

データ・ディクショナリ・キャッシュ, 8-13, 8-14

障害

ALERT ファイル, 7-2

メディア, A-24

障害検出

- Cluster Manager による, 9-14

障害保護妥当性チェック, 9-7

冗長性計画, 9-4

初期化パラメータ

廃止, A-23

新規、機能

プライマリ / セカンダリ・インスタンス, A-2

新機能, ix

親和性

- パラレル・プロセッサ, A-17

す

スケーラビリティ, 1-3

- 4つのレベル, 8-10
- リリース7.3で拡張, A-13
- アプリケーション, 8-10, 8-12
- オペレーティング・システム, 8-11
- 可能性, 8-2
- 共有メモリー・システム, 8-11
- 相対的, 8-7
- 定義, 8-3
- データベース, 8-12
- ネットワーク, 8-13
- ハードウェア, 8-11
- スピードアップ
 - 定義, 8-4
- スピードダウン, 8-5
- スレッド
 - グループ番号, 7-4
 - 例, 7-3

せ

- 制御ファイル
 - MAXLOGHISTORY, 7-5
- 制限事項
 - オフライン表領域, 7-6, A-20
 - キャッシュされた順序, 8-14
 - 遅延ロールバック・セグメント, A-21
 - ファイル操作, A-19, A-21, B-3
 - リカバリ, A-24
- 静的ハッシング
 - ロック・マスター化構造, 3-5
- セグメント
 - ヘッダー、競合, 7-14
 - ヘッダーに対する競合, 7-14
 - ロールバック・セグメント, 7-6
- 絶対ファイル番号, 7-3
- 専用サーバー
 - プライマリ / セカンダリ・インスタンス, 9-18

そ

- 相対ファイル番号, 7-3
- 挿入
 - パフォーマンス, 7-19
- ソート拡張, A-12
- ソート領域, A-12
- 素数, A-8

た

- 帯域幅, 2-4, 2-5, 8-11
- 対称型マルチプロセッサ, 8-7, 8-11
- 多重 REDO ログ・ファイル, 7-3
 - 例, 7-4
- 単一共有モード, 4-8
- 単一の致命的な障害箇所
 - 高可用性での回避, 9-2

ち

- 遅延ロールバック・セグメント, A-21
- 超並列システム, 2-7, 7-21
 - アプリケーション・プロファイル, 8-7

て

- ディクショナリ・キャッシュ, 4-2
 - ロック, 4-7
- ディクショナリ・キャッシュ・ロック, 4-7
- 停止
 - 原因, 9-3
 - 停止時間
 - 予定外の, 9-3
 - 予定された, 9-3
- ディスク
 - 競合, 7-2, 7-7
 - ブロックの書込み, 1-4, 5-22
 - ブロックの読み込み, 1-4
 - ロールバック・セグメント, 7-7
- ディスク・サブシステム, 3-6
- データ・ウェアハウス, 8-5
- データ・ディクショナリ
 - オブジェクト, 4-2
 - 行キャッシュ, 4-2
 - 順序キャッシュ, 8-14
- データ・ファイル
 - PCM ロックに未指定, 5-20
 - インスタンス・リカバリ, 7-3
 - オフライン化またはオンライン化, A-20
 - 改名, A-20
 - 共有, 7-2
 - 高可用性でのアクセス, 9-6
 - 最大数, B-3
 - 削除, A-20
 - 追加, A-20

- ディスク競合, 7-2
- 表ごとの複数ファイル, 7-21
- 表領域, A-20
 - ブロックへのロックのマッピング, 5-20
 - リカバリ, A-24
- データベース
 - アーカイブ・ログ・ファイル数, 7-5
 - インスタンス数, 7-21
 - エクスポートおよびインポート, B-2
 - スケーラビリティ, 8-12
 - バックアップ, 1-5
- データベース・インスタンス登録
 - Connect-Time Failover, 8-10
 - クライアント・ロード・バランス, 8-10
- データベース・オブジェクトの削除
 - 表領域, A-20
- デッドロック検出, 3-5

と

- 透過性
 - 定義, 1-3
- 透過的アプリケーション・フェイルオーバー
 - 使用, 9-10
 - 定義, 9-9
- 同期化, 4-2
- 統合分散ロック・マネージャ
 - 内部化, A-9
- 同時実行性
 - インスタンスの最大数, 7-21
 - 順序, 8-14
- トランザクション
 - 異常終了, 7-6
 - オフライン表領域, 7-7, A-21
 - 行ロック, 1-4, 4-12
 - 更新, 1-4, 7-10
 - コミットされたデータ, 1-4
 - 順序番号, 8-13
 - 挿入, 7-10
 - 同時, 1-4, 4-9, 4-10
 - 分離, 4-12
 - リカバリ, A-14
 - ロールバック, 7-6
 - ロールバック・セグメント, 7-7, A-21
 - ロック, 4-5, 4-7
- トランザクション・リカバリ, 9-16

に

- 二重ポート・コントローラ, 8-11

ね

- ネットワーク・パフォーマンスの向上
 - クライアント要求のランダム化による, 8-10

の

- ノード
 - キャッシュ一貫性, 4-9
 - 高可用性, 9-6
 - 障害, 1-3
 - 定義, 2-2
 - ハードウェア, 2-1
- ノード間通信, 4-2

は

- バージョン、Oracle
 - アップグレード, A-1
 - 互換性, A-19
- パーティション化データ
 - PCM ロック, 7-21
 - 空きリスト, 7-10, 7-21
 - ~を持つアプリケーション, 8-6
 - データ・ファイル, 7-2
 - 表データ, 7-21
 - ロールバック・セグメント, 7-6, 7-7
- ハードウェア
 - スケーラビリティ, 8-11
 - パラレル処理, 2-1
 - 要件, 3-2
- 廃止されたパラメータ, A-22, A-23
- 排他的 PCM ロック, 4-12
- 排他モード, 5-9
 - 互換性, B-2
 - 表領域のオフライン化, A-20
 - ファイル操作に必要, A-19
- バックアップ
 - エクスポート, B-2
 - オフライン, 1-5
 - オンライン, 1-5
- バックグラウンド・プロセス
 - インスタンス・ロックの保持, 4-4

- ハッシュ PCM ロック, 5-4
- ハッシュ・クラスタ
 - 空きリスト, A-22
- ハッシュ・ロック
 - 使用する場合, 5-32
- ハッシング
 - 静的、ロック・マスター化構造, 3-5
- バッファ・キャッシュ
 - I/Oの最小化, 1-4, 4-9
 - 一貫性, 4-9
 - ディスクへの書込み, 1-4
 - 分散ロック, 8-6
- バッファ・キャッシュ管理, 1-4
- バッファ状態, 5-24, 5-25
- パフォーマンス
 - Oracle8 機能, 1-4
 - アプリケーション, 7-19
 - 順序のキャッシュ, 8-14
 - 順序番号, 8-14
 - 挿入および更新, 7-19
 - ロールバック・セグメント, 7-7
 - ロック・マスター化, 3-5
- パラメータ
 - 記憶域, 7-7
 - データベースの作成, 7-21
 - 廃止, A-23
- パラレル・キャッシュ管理, 4-8
 - 例, 4-10
- パラレル・キャッシュ管理ロック
 - NULL, 4-12
 - インスタンス LCK プロセスによって所有, 5-17
 - 解放, 4-12
 - 周期性, 5-22
 - 取得, 4-12
 - 順序, 8-13
 - 相対数, 4-13
 - 動作方法, 5-31
 - 排他, 4-12
 - 複数インスタンスによって所有, 5-17
 - ユーザー制御, 4-13
 - 読み込み, 4-12
- パラレル実行
 - 実行プロセス, 8-12
 - プロセッサの親和性, A-17
- パラレル処理
 - 有利な場合, 8-5
 - 定義, 1-2

- ハードウェア, 2-1
- パラレル・データベース
 - 可用性, 1-3
- パラレル・トランザクション・リカバリ
 - リリース 8.1 で変更, A-4
- パラレル・プロセッサの親和性, A-17
- パラレル・モード
 - 順序の制限事項, 8-14, B-3
 - ファイル操作の制限事項, A-19, A-21, B-3
 - リカバリの制限事項, A-24
- 半共有排他モード, 5-9
- 半共有モード, 5-9
- 番号ジェネレータ, 8-13

ひ

- 非 PCM ロック
 - DML ロック, 4-7
 - IDLM 容量, 4-13
 - エンキュー, 4-4
 - 概要, 4-6
 - 相対数, 4-13
 - タイプ, 4-5
 - ディクショナリ・キャッシュ・ロック, 4-7
 - トランザクション・ロック, 4-7
 - 表ロック, 4-7
 - マウント・ロック, 4-8
 - ユーザー制御, 4-13
 - ライブラリ・キャッシュ・ロック, 4-7
- 非固定モード、ロック要素, 5-19
- 非対称マルチプロセッシング, 8-11
- 非同期トラップ, 5-27, 5-29, 5-30
- 表
 - PCM ロック, 7-21
 - エクステントの割当て, 7-24
 - 競合, 7-6
 - 初期記憶域, 7-25
 - 表領域, 7-6
 - ロック, 4-4, 4-13
- 表領域
 - アクティブ・ロールバック・セグメント, 7-6
 - オフライン化, 7-6, A-20, A-21
 - 削除, A-20
 - 作成, A-20
 - データ・ファイル, A-20
 - バックアップ, 1-5
 - 表, 7-6

リカバリ, A-24
ロールバック・セグメント, 7-6
表領域の作成, A-20
表ロック, 4-7

ふ

ファースト・スタート・リカバリ, 9-5, 9-16
ファースト・スタート・ロールバック, 9-16, A-4
ファイル
ALERT, 7-2
REDO ログ, 1-5, 7-3, A-20
REDO ログのアーカイブ, 1-5
エクスポートされた, B-2
改名, A-20
サイズ, 5-20
最大数, B-3
削除, A-20
制御ファイル, 7-5
制限された運用, A-19
追加, A-20
番号、絶対, 7-3
番号、相対, 7-3
ファイルの改名
REDO ログ・ファイル, A-20
RENAME FILE オプション, A-20
ファイルの追加, A-20
ファイングレイン・ロック, 8-6, A-11
1 ロック要素あたり 1 ブロック, 5-15
DBA ロック, 5-15
フェイルオーバー, A-13
期間, 9-9
基本, 9-8
サーバー側, 9-12
接続時間, 8-10
定義, 9-2
透過的アプリケーション・フェイルオーバー, 8-10
ホストベース, 9-12
フォールト・トレラント, 3-5
不均一ディスク・アクセス, 2-7
不均一メモリー・アクセス
定義, 2-3
複数共有モード, 4-8
複数バージョン読取り一貫性, 1-4
部門別アプリケーション, 8-7
プライベート・ロールバック・セグメント
指定, 7-8

取得, 7-6
プライマリ / セカンダリ・インスタンス
定義, 9-18
プロセス
Oracle Parallel Server, 6-2
プロセス空きリスト
セグメント・ヘッダーの ping, 7-14
プロセス間通信 (IPC)
記述, 6-5
プロセッサの親和性
パラレル実行, A-17
ブロッキング AST, 5-29
ブロック
インスタンスとの対応付け, 7-24
競合, 7-7, 7-21
クリーンアウト, A-16
遅延ロールバック, 7-7, A-21
ディスクへの書込み時, 1-4, 5-22
複数コピー, 1-4, 4-9
ブロックの動的割当て, 7-25
ブロック・レベル・ロック, 4-13
ブロック・サーバー・プロセス
定義, 6-5
プロテクト書込みモード, 5-9
分散ロック
行キャッシュ, 4-2
合計数, 4-2
順序, 8-13
ロールバック・セグメント, 7-7
分散ロック・マネージャ, A-9
LCKn プロセス, 5-22
LMON, 9-14
機能, 3-4
キュー, 5-27
グループ・ベースのロック, 5-33
使用を最小化する, 4-9
非 PCM ロック容量, 4-13
フォールト・トレラント, 3-5
分散アーキテクチャ, 3-5
リソース共有, 5-22
ロック要求を操作する, 5-28
分散ロック・マネージャ (DLM)
説明, 3-4

へ

- 平均障害間隔時間
 - 定義, 9-2
- 平均リカバリ時間 (MTTR)
 - 定義, 9-2
- 変換キュー, 5-28

ほ

- ホスト, DLM, 5-22
- ホストベース・フェイルオーバー, 9-12

ま

- マウント・ロック, 4-8
- 待ち時間, 2-4, 8-11
 - インターコネクト, 2-7
- マルチスレッド・サーバー, 5-33
 - プライマリ / セカンダリ・インスタンス, 9-18
- マルチティア・アプリケーション環境, 9-9

め

- メッセージ
 - ALERT ファイル, 7-2
 - ファイルへのアクセス, 7-2
 - 分散ロック・マネージャ, 5-22
- メディア障害, A-24
 - ファイルへのアクセス, 7-2
 - リカバリ, A-24
- メモリー
 - DLM 要件, 5-34
 - SGA, 8-14
 - キャッシュ, 1-4
 - キャッシュ・データ, 1-4
- メモリー・アクセス, 2-2
- メモリーマップされた IPC
 - Oracle Parallel Server での使用方法, 2-4

も

- モード
 - PCM ロック, 4-12
 - アーカイブ, 1-5
 - 非互換, 5-27
 - ロック互換性, 5-26

ゆ

- 有効ビット、ロック要素, 5-19
- ユーザー・プロセス
 - 空きリスト, 7-10, 7-21
- ユーザー・モードのプロセス間通信
 - Oracle Parallel Server での使用方法, 2-4
- ユーザー・レベル DLM, 5-34
- ユーティリティ、Oracle
 - エクスポート、インポート, B-2
- ユニプロセスサ
 - 定義, 2-2

よ

- 予定外の停止時間, 9-3
- 予定された停止時間, 9-3
- 読込み専用アクセス, 1-4, 4-12
 - アプリケーション, 8-6
 - 読込み PCM ロック, 4-12
- 読込みロック・モード, 4-12
- 読取り一貫性
 - 複数バージョン, 1-4
 - ロールバック情報, 7-6

ら

- ライブラリ・キャッシュ・ロック, 4-7
- ラッチ, 4-3, 4-8
- ランダム・アクセス, 8-6

り

- リカバリ
 - インスタンス, 9-15
 - キャッシュ, 9-16
 - 制限事項, A-24
 - 遅延トランザクション, A-14
 - ファイルへのアクセス, 7-2, 7-3
 - メディア障害, 7-2, A-24
 - ロールバック, 7-6
- リスナー
 - Connect-Time Failover, 8-10
 - 透過的アプリケーション・フェイルオーバー, 8-10
 - ロード・バランス, A-5
- リスナー間での要求のランダム化, 8-10

- リソース
 - 持続, 3-6
 - データベース, 4-9
- リソース・マスター化, 3-5
- リモート I/O, 8-11
- 領域
 - 空きブロック, 7-10, 7-25
 - 空きリスト, 7-10

ろ

- ローカル I/O, 8-11
- ローカル・ロック, 4-3
- ロード・バランス, A-15
 - クライアント・ロード・バランス, 8-10
- ロールバック
 - 行ロック, 4-12
 - ロールバック・セグメント, 7-6
- ロールバック・セグメント
 - SYSTEM, 7-6
 - 説明, 7-6
 - 表領域, 7-6
 - オンライン, 7-6, A-15
 - 競合, 7-6, 7-7
 - グローバル定数パラメータ, 7-7
 - 遅延, 7-7, A-21
 - パブリックおよびプライベート, 7-7
 - 複数, 7-6
 - 分散ロック, 7-7
- ロールバック・セグメントの取得
 - 初期化パラメータ, 7-7
- ログ・スイッチ, 7-5
 - 強制, A-21
- ロック, 1-4
 - DLM によるオーナーシップ, 5-33
 - DML, 4-4, 4-7
 - PCM ロック, 7-21
 - エンキュー, 4-3
 - 行, 4-7, 4-12
 - 境界, 7-26
 - 行キャッシュ, 4-2
 - 行ロックの独立性, 4-12
 - グループ・ベース, 5-33
 - グローバル, 4-4
 - コスト, 5-6
 - 再マスター化, 9-15
 - ディクショナリ・キャッシュ, 4-7

- トランザクション, 4-7
- 非 PCM, 4-5
- 表, 4-4, 4-5, 4-7, 4-13
- プロセス所有, 5-33
- 変換, 5-30
- マウント・ロック, 4-8
- モード互換性, 5-26
- モード、バッファ状態, 5-24
- 要求、DLM による操作, 5-28
- ライブラリ・キャッシュ, 4-7
- ラッチ, 4-3
- ローカル, 4-3
- ロールバック・セグメント, 7-7
- 非 PCM, 4-4
- ロック要素
 - LRU リスト, 5-19
 - 空き, 5-19
 - 名前, 5-15
 - 非固定モード, 5-19
 - 有効ビット, 5-19
 - ロックへの対応, 5-14

わ

- 割当て
 - PCM ロック, 5-20, 7-21
 - 空き領域, 7-24
 - 自動, 7-25
 - 順序番号, 8-14