耐障害システムのためのデータベース接続層の拡張による柔軟な複製機構

中村暢達[†,††] 藤山健一郎[†]
河合栄治[††] 砂原秀樹[††]

近年，社会的にもITシステムへの依存が大きくなり，障害発生によるシステムダウン時にも，バックアップサイトが同一のITサービスを継続するような障害復旧技術の重要性がますます高くなりつつある．本論文では，処理性能，復旧時間（RTO），および復旧ポイント（RPO）などの障害復旧に関する多様なサービス要件に応じて柔軟にデータを複製する方式を提案する．高い柔軟性を実現するために，データベース接続ライブラリにおいて，アプリケーションのデータベースアクセスを監視し，あらかじめ登録されたルールと照合し，複製方式を制御する．具体的には，複製および復旧の要件から，データベースアクセスとして以下の4種類の複製方式を用意する：1) 複製する必要のないアクセス，2) 非同期で複製すればよいアクセス，3) 非同期ではあるが，アクセス順序は同一でなければならないアクセス，4) 同期複製する必要のあるアクセス．各アクセスに応じた最適な複製方式を選択することで，遠隔地へ複製処理を行う際の性能劣化を抑えることが可能となる．実験においては，遠隔通信環境下でも実適用可能な処理性能で複製方式制御が動作することを確認した．

A Flexible Replication Mechanism with Extended Database Connection Layers for Disaster Recovery System

Nobutatsu Nakamura,[†,††] Ken'ichiro Fujiyama,[†] Eiji Kawai[††] and Hideki Sunahara[††]

It is vital to achieve a disaster recovery system that allows a backup site to take over primary site IT services while the primary site is down. We propose a flexible replication mechanism based on service requirements such as system performance, recovery time objective (RTO), and recovery point objective (RPO). For high flexibility, the mechanism controls the replication schedule by monitoring the application database accesses in the database connection library and matching the accesses with previously registered access patterns. In the proposed system, the database accesses are classified into four types: 1) accesses without replication, 2) accesses with asynchronous replication, 3) accesses with asynchronous and order-assured replication, and 4) accesses with synchronous replication. By choosing a suitable replication method corresponding to each access, performance degradation caused by replication is largely alleviated. In our experiments, we confirmed that the proposed mechanism outperformed other existing mechanisms, especially in situations with network delays and packet losses.

1. はじめに

近年，ITシステムがビジネスに不可欠となり，災害，破壊活動などのためにシステムが停止しても，別のサイトにおいて，同じITサービスを継続させることが求められている．金融やキャリア向けのITシステムにおいては，高いコストをかけて，専用の耐障害システムが構築されていることが多い．しかし，多くのITシステムでは，テープなどの別メディアにバックアップを作成し，災害・障害発生時には，バックアップからデータを復元するのが一般的であり，時間と労力がかかる作業である．そこで，通常時の運用サイト（プライマリサイト）が停止した場合に，別サイト（バックアップサイト）において同一サービスを迅速に引き継ぐ復旧方式がいくつか提案されている．特に，プライマリサイトとバックアップサイトとの間でデータを同期して更新するデータ複製を行い，複製データを用いてサービスを継続する方式が注目されている．

† 日本電気株式会社
NEC Corporation
†† 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

既存のデータ複製技術は大きく Primary Copy と Update Anywhere の 2 つに分けられる[20]．一般に Update Anywhere は高信頼であり，Primary Copy は高性能とされる．

本論文では，システムの処理性能と信頼性とのバランスを制御可能とし，高い柔軟性を持つデータ複製技術方式を提案する．提案方式は，信頼性の高い Update Anywhere 方式を基にするが，すべての処理を 1 つ 1 つ同期的に処理することはせず，処理状況（処理の意味）に応じて，複製方式を可変とするというものである．信頼性は，通常 RTO および RPO と呼ばれる復旧性能が指標となる．RTO（Recovery Time Objective）とは，縮退運転のように最大性能での再開でなくても，とにかくサービス再開するまでの時間である．RPO（Recovery Point Objective）とは，サービスが再開される時点のことで，最後に有効なバックアップを取得した時点であることが多い．RTO=0，RPO=0 というのが理想値であり，障害が顕在化しないことを意味する．処理性能と信頼性（RTO，RPO）とは，通常トレードオフの関係にあり，耐障害機構としては，バランス良く，多様なサービスやアプリケーションに適応することが求められる．

今日，多くのシステム管理者は費用対効果を重要視する傾向にあり[4]，遠隔通信に，専用回線よりインターネットを利用することが多い．このため，システム設計においてネットワーク遅延とパケット損失等を考慮する必要がある．提案方式では，各処理要求の意味に応じて複製方式を制御し，複製するデータサイズそのものを削減することで，これらの影響を小さくし，インターネットを利用することを可能とする．さらに，複製するデータサイズが小さくなれば，ネットワークだけでなく，CPU 性能，メモリなどの要件が低くなり，ハードウェアのコストを抑えることも可能である．

以下，本論文は次のような構成で議論を進める．2 章では，既存のデータ複製方式を鑑み，複製要件を整理する．3 章で提案するシステム構成の概要を示し，4 章で提案する複製機構を詳細に説明し，5 章でディザスタリカバリへの適用を説明する．6 章では，性能評価のための実験および結果を示す．7 章では，関連技術について概説し，最後 8 章でまとめを述べる．

2. 複製処理要件

停止したプライマリサイトのサービスをバックアップサイトで復旧するためには，バックアップサイト側でプライマリサイトと同じデータを保持する必要がある．そして，いかにサービスを停止することなく継続してサービスできるかは，プライマリサイトとバックアップサイトがどのくらい正確に同期できているかに依存する．本章では，既存のデータ同期複製方式を説明し，その特徴を議論する．

サービスは，**図 1** に示すように，大きくクライアント，アプリケーション（AP）サーバ（Web サーバを含む），データベース（DB）サーバ，ストレージという経路で処理される．データ複製のためには，この経路途上のいずれかの位置で，データもしくは処理コマンドを二重化し，プライマリとバックアップの両方のサイトで，同一のデータ処理を行う．サイト間で整合性を保持するには，両サイトの処理内容および順序は同一である必要があり，もし異なっていれば，データの状態は異なる可能性が高い．

ここで，上位レイヤ（クライアント側）でのアクセス要求は，低レイヤ（ストレージ側）のアクセス要求につねに同一に展開されるとは限らない．たとえば，ファイル A の更新要求があった場合，オペレーティングシステムの実装，システム負荷，およびネットワーク条件などにより，物理的にストレージのどの位置にデータが更新されるかは確定的でない．しかしながら，サービスの観点から見れば，両サイトのストレージが，このような物理的なデータ状態まで同一である必要はなく，論理的に同一であれば十分である．

上位レイヤで複製する場合，ほかに処理順序が問題となる．あるクライアントからアクセス要求 A が発行され，ほぼ同時に別のクライアントからアクセス要求 B が発行された場合，プライマリサイトでは A → B の順に処理するが，バックアップサイトでは B → A の順に処理するかもしれない．そのために，両サイトでの状態が異なる可能性があり，処理順序が同一となるように保証することが必要である．

同一処理順序を保証するためには，ストレージに直接アクセスする，もしくは DB サーバとのトランザクション処理を行うこととなる．

ストレージレイヤにおける複製：一部のストレージ製品には，装置内部にデータ複製機能を有し，広帯域の専用線で接続された異なる場所のストレージ装置との間で同期複製を行うものがある．この場合，直接ストレージ装置に対する処理データを複製しており，最もシンプルな複製方法であるといえる．しかしながら，広域ネットワークのスループットはストレージへのアクセス I/O スループットより低いため，複製処理は大きな性能劣化を引き起こす．上位レイヤでは 1 つのコマンドであっても，下位レイヤでは，OS やストレージの管理・制御のために，高頻度に大量のデータをや

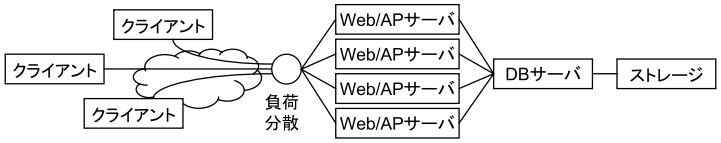

図 1 サービス処理の流れ
Fig. 1 Service workflow overview.

りとりしていることもあり，遠距離通信のオーバヘッドによる性能劣化の影響を受けやすい．

また，ストレージレイヤの複製では，復旧処理の信頼性にも問題がある．バックアップサイトで保持されているデータは，再開可能な状態で保存される必要があるが，その保証が困難なことである．たとえば，データの更新途中で，プライマリサイトの障害が発生した場合，その更新途中のデータは，最後まで更新されるか，更新前に戻るかのいずれかの状態となることが少なくとも必要である．このような原子性の保証と呼ばれるデータ管理は，通常 OS 層より上位で実現される．

DB レイヤにおける複製：一方，DB サーバは ACID 性[9]，つまり Atomic（原子性），Consistent（一貫性），Isolated（分離性），Durable（存続性）を実現するように構成されたシステムであり，原子性は保証されている．また，DB レイヤのアクセス要求（トランザクション）は，ストレージレイヤよりは抽象化されたレベルのコマンドであり，一般に通信データサイズはより小さく，通信オーバヘッドの影響を受けにくい．そのため，DB レイヤで実現するデータ複製技術は数多い．またトランザクションは，SQL などのテキスト形式であるため，データ処理・制御の柔軟性は高い．

既存の DB レイヤでの複製技術は，DB サーバ内部の複製と外部の複製に分けられる[7]．いくつかの DB 製品においては，DB 内部型の複製機能を提供しており，ベンダ独自のプロトコルで，トランザクションを複製し，バックアップ DB サーバに通信することで，複製機能を実現している．しかし，複製機能はハイエンドのエンタープライズバージョンに限定されていた．

最近，複製機能はいくつかのオープンソースの DB 製品にも実現されてきているが，まだ機能的に十分であるとはいえない．たとえば，MySQL Cluster[15] と Postgres-R[11] は複製機能を追加するために，プログラムソースコードを修正する必要がある．PGCluster[18] は修正なしで複製機能を追加することが可能だが，オリジナルの PostgreSQL データベースエンジンを変更する必要がある．

表 1 複製要件に対する各複製方式の特徴
Table 1 Replication features for relication requirement.

複製位置	ストレージ	DB（内部）	DB（外部）
導入の容易さ	○	×	×
原子性保証	△	○	○
サーバ負荷	○	×	△
通信帯域	×	○	○
冗長体制復旧	○	△	△
柔軟性	×	△	○

そのほかに負荷の問題がある．DB サーバは，分散構成では整合性を確保するオーバヘッドが大きいため，単一サーバで構成されることが多い．そのため，DB サーバには高性能のサーバマシンを用いるが，サーバマシンの追加による処理性能増強ができないので，想定以上の処理で DB サーバがボトルネックとなることも多い．したがって，DB サーバの負荷が問題となるシステムの場合，DB サーバ内部に複製機能を追加することは，不適切である．

DB サーバ外部での複製方法としては，Pgpool[23]，C-JDBC[3] などの技術がよく知られている．Pgpool は，PostgreSQL へのアクセスを複製するプロキシサーバをシステムに追加し，また C-JDBC においても，C-JDBC コントローラと呼ばれるモジュールをシステムに直列追加する．このような追加モジュールは，AP サーバと DB サーバのネットワーク上にあっても，またいずれかのサーバ内にあってもよい．ただし，DB サーバ外部での複製を実現する際，このような追加モジュールが，単一障害点とならないように，かつ性能ボトルネックとならないようなシステム構成とすることが必要である．

また，DB レイヤにおける複製では，障害後の冗長体制への復旧に際し，DB を再構築するためのログを保存する大量のストレージを要し，またログを再投入処理するので復旧に長時間を要するというような課題がある．そのため，データベースを一時停止し，スナップショットを取得，それを復元して冗長体制を復旧する方式が用いられることも多い．

以上の議論をまとめると，**表 1** のようになる．アプリケーションの複製要件に応じて，適した複製方式を選択すればよく，インターネットの利用，柔軟性の

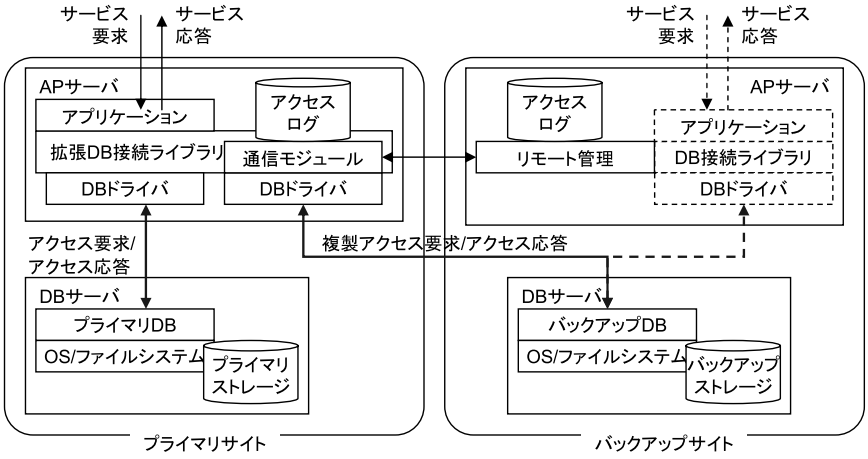

図 2 提案システム構成の概要
Fig. 2 Proposed system overview.

高い複製を目的とする場合，DB サーバ外部での複製が適する．さらに，単一障害点を新たに設けない，アプリケーションのコード修正が不要などの観点から，DB 接続ライブラリを置換する方式を提案する．次章でその詳細を述べる．

3. データ複製システム

DB 接続ライブラリとは，AP サーバにおいて，DB サーバとの接続手続を仮想化することで，DB サーバの種類に依らない統一的な API を提供する．ODBC[6]，JDBC[14]，ADO.NET[16] などが DB 接続ライブラリとしてよく知られている．DB 接続ライブラリは，アプリケーションもしくはサーブレットが発行する DB 製品非依存のアクセス要求を受け取り，DB 製品固有のアクセス要求に変換し，DB サーバに送信する．DB サーバからアクセス応答を受け取り，それを DB 製品非依存のアクセス応答に変換してアクセス要求元に返す．DB 製品固有の処理は，ベンダから提供される DB ドライバが担う．

本提案では，DB 接続ライブラリを拡張し，アプリケーションのアクセス要求を複数の DB サーバに送信し，またそれらからの応答を受け取り，1 つの応答のみをアプリケーションに返すようにシステムを構成する．

図 2 に提案するシステム構成の概要を示す．システムは，プライマリサイトとバックアップサイトからなり，両サイトは，AP サーバと DB サーバを含む．プライマリサイトの AP サーバは，サービス要求の受信とサービス応答の返信を行うアプリケーション，サイト間通信モジュールを備えた拡張 DB 接続ライブラリ，ローカルの DB サーバと接続する DB ドライバ，遠隔のバックアップサイトの DB サーバと接続する DB ドライバからなる．サイト間通信モジュールは，アクセスログ（アクセス要求・応答のログ）をバックアップサイトに送信することと，DB ドライバを介して，バックアップサイトの DB サーバとデータアクセス要求・応答を通信することを担う．なお，アクセスログの保存は，シーケンシャルに追記する処理であり，通常のデータベース処理と比べ，高速に処理が可能であることが知られており[21]，本論文ではその詳細な説明を省く．

バックアップサイトの AP サーバには，災害時に有効となるアプリケーション，バックアップサイト内の DB 接続ライブラリおよび DB ドライバがスタンバイされている．リモート管理は，通常時はプライマリサイトからアクセスログを受け取り，保存する．障害発生時には，復旧処理を担う．プライマリサイトとバックアップサイトの DB サーバは，データベース，OS（ファイルシステム），ストレージから構成される．

4. データ複製制御機構

4.1 データ複製方式

プライマリサイトの AP サーバ内の標準 DB 接続ライブラリを複製機能を備える拡張 DB 接続ライブラリに置き換えることで，データ複製システムを実現する．拡張 DB 接続ライブラリは，標準の DB 接続ライブラリの API と互換性を保つので，アプリケーションのソースコードを修正する必要はない．また，複数のデータベース接続を行い，接続の 1 つは，DB ドライバを介して，プライマリ DB に接続し，別の接続は通信モジュールにリンクする．通信モジュールは，DB

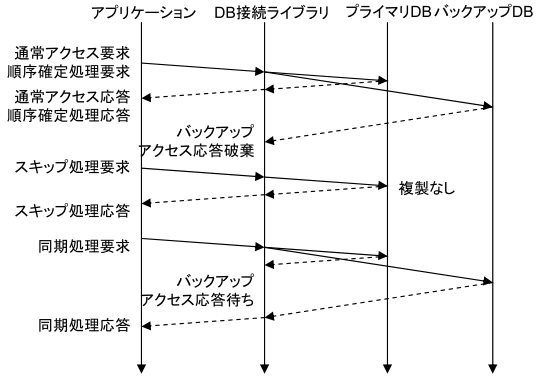

図 3 複製処理のアクセス手順
Fig. 3 Replication access sequences.

ドライバを介して，バックアップ DB に接続する．両サイトの DB サーバが論理的に同一の状態を保持するように，プライマリ DB に送信するアクセス要求を，同時にバックアップ DB にも送信する．

一般に，データベースシステムを利用する場合，connect（DB への接続），execute（データ処理の実行），commit（データ処理の確定），close（DB との接続断）の順にトランザクション処理が進む．各トランザクションは，複数のデータベースアクセスを含むが，アクセスの種類によっては，バックアップ DB 側で同時に処理する必要はない場合もある．一連のデータベースアクセス，時間，処理数などをアクセス状況と呼び，そのアクセス状況に応じて，異なる複製方式で処理を実行する制御機構を設ける．本論文では，通常処理，スキップ処理，順序確定処理，同期処理の 4 種類の複製方式を用意した．4 種類の複製方式のアクセス手順を**図 3** に示す．

通常処理では，DB 接続ライブラリは，アプリケーションからのアクセス要求を受け，プライマリ DB とバックアップ DB の両方にアクセスする．プライマリ DB からアクセス応答を受け取った後に，アプリケーションに返す．バックアップ DB からのアクセス応答も受け取るが，そのアクセス応答は破棄される．

スキップ処理では，DB 接続ライブラリは，アプリケーションからのアクセス要求を受け，プライマリ DB のみにアクセスし，バックアップ DB にはアクセスしない．

順序確定処理では，基本的に通常処理と同じであるが，処理順序を確定して処理を進める点で異なる．複数の AP サーバを含むシステムの場合，プライマリ DB とバックアップ DB とで，トランザクション処理の順序が異なる可能性があり，両 DB で処理順序がつねに同じになるように保証する場合に適用される．順序確定の 1 つの方法としては，各トランザクションの発行順序を一元管理するセマフォサーバを用意する手法が考えられる．各 AP サーバは，トランザクション発行前に，セマフォサーバに許可要求し，セマフォサーバは，許可要求をキューに加える一方，キューの先頭の許可要求に対し，要求元の AP サーバに発行許可を与える．許可を受けた AP サーバは，トランザクションを発行し，発行したことをセマフォサーバに通知する．セマフォサーバは，発行済みの通知を受けると，次のキューの許可要求を処理する．

同期処理では，プライマリ DB とバックアップ DB の両方に，同時にアクセスするようにアクセスを制御する．さらに両方のデータベースから，アクセス応答を受け取り，そのアクセス応答をアプリケーションに返す．

4.2 データベースアクセス状況

本提案方式では，順序確定処理や同期処理が必要なシステム状態およびイベントログをアクセス状況としてあらかじめ登録しておき，実際にアクセスログなどを監視して得られたシステム状態およびイベントログと，登録されたアクセス状況とを照合して合致した場合に，順序確定処理，あるいは同期処理を行う．アクセス状況には，データベースへのアクセスの種類，操作テーブルの種類，セッションのユーザ ID，グループ ID，アクセス元（IP アドレス），日時などがある．

順序確定処理，同期処理を高頻度で実行すれば，システムの信頼性は向上するものの，順序確定処理は並列的に処理できないため，また同期処理は遠隔地のバックアップ DB の更新確認を待つために，システム処理性能を劣化させる．登録するアクセス状況を変更することで，順序確定処理，同期処理の頻度を管理し，システム処理性能と RTO，RPO の信頼性とのバランスを柔軟に制御できる．システムによっては，通常時の処理性能を重視し，順序確定処理，同期処理を厳密に行わず，RTO，RPO の信頼性を犠牲にするという設定も可能である．

DB アクセス状況は多様であり，また各アクセスの意味の理解は困難であることから，アプリケーションプログラムの詳細を知らない運用者が，アクセス状況と複製方式の関連付けを直接記述することは困難な作業である．実際の運用場面においては，システム構築ベンダがルールセット，もしくはアプリケーション種類別のテンプレートを提供するものとする．

4.3 複製処理時の障害

本節では，ディスク障害など日常発生する障害に対

する処理について述べる．具体的には，いずれかの DB の応答がなくなる場合と，いずれかの DB がエラーを返す場合について議論する．ここでは，本提案で独自拡張を行っている複製制御機構に関連する障害についてのみ言及し，より大規模な障害への対応については 5 章で述べる．

いずれかの DB サーバで障害が発生し，アクセス応答がない場合，DB 接続ライブラリにおいて，その DB サーバとの通信はタイムアウトし，片方の DB サーバのみで処理を継続する．障害発生中は，その DB サーバへのアクセス要求をセマフォサーバに保存しておく．AP サーバが 1 つの場合には，アクセス要求の順番を考慮する必要がないので，アクセス要求は AP サーバ内にローカルに保存すればよい．DB サーバが復旧した際には，そのアクセス要求を使って，障害発生中のデータベース処理を再現する．すべての保存したアクセス要求を処理した時点で，サービスを一時停止し，通常の複製処理構成に戻す．

障害発生中のデータベース処理を再現する間も，片方の DB サーバでサービスを継続し，アクセス要求を保存するが，再現処理が遅いと通常の複製処理構成に復帰できない．このような場合には，AP サーバ側でサービスを制限するなどの対処を行う必要がある．

障害が長時間になると，アクセス要求を保存するサイズが大きくなり，ディスク容量が不足，また障害発生中のデータベース処理の再現に長時間を要するなどの問題がある．その場合には，チェックポイント，スナップショットなどのデータの保存・復元技術を利用する．

いずれかの DB サーバで障害が発生し，エラーを含むアクセス応答があった場合，2 つのアクセス応答だけでは，どちらが正しいアクセス応答かは分からない．本提案の同期処理では，どちらかのアクセス応答を使って処理を進めることはせず，アプリケーションにエラーを返し，両 DB はロールバックする．同期処理以外では，異なる応答があってもプライマリ DB サーバの応答のみで処理は進む．異なる応答があった場合に処理を進めては不都合となるアクセス要求には，同期処理を行うことが不可欠となる．

上述した方式は，正しい応答が含まれていてもエラーとするため，処理性能は劣化する．もし，異なる応答の中から，確からしい応答を使って処理を進めるには，3 つ以上の DB サーバを用意し，多数決で決定するなどの方式がある．

5. ディザスタリカバリ

本提案システムでは，自然災害による地域的な停電，ネットワークの停止など，プライマリサイト全体が停止する場合，バックアップサイトにおいて，プライマリサイトで提供していたサービス環境を引き継ぎ，バックアップ DB を使って，同一サービスを提供する．本章では，その手順を詳細に述べる．

5.1 フェイルオーバ

災害が発生し，AP サーバが停止した場合，クライアントとのセッションは失われ，バックアップ DB サーバにあった未確定のアクセス要求は破棄される．AP サーバの製品には，クライアントの処理状態を保存し，その状態をバックアップサイトの AP サーバに引き継き，DB サーバ側も，引き続いてアクセス要求処理を継続する機能を持つものもある．しかし，クライアントの処理状態を引き継がない方がシステムの実装は容易であり，オーバヘッドも少ないことから，AP サーバの状態保存はいまだ利用は少ない．

処理状態を引き継がない場合でも，確定処理の途中で災害が発生した場合には，クライアントとバックアップ DB サーバとで確定，未確定の状態が異なる可能性がある．このような不整合の回避については後述する．

バックアップサイトで，サービスを継続するフェイルオーバは，通常システム管理者の判断により開始する．まず，これまでのプライマリサイトのサービスをネットワークから切り離されていることを確認し，バックアップサイトで，サービスしていた URL，IP アドレスを引き継ぎ，AP サーバを起動する．さらに，AP サーバをバックアップ DB に接続し，サービスを開始する．

フェイルオーバ後に，プライマリサイトが復旧した場合，もしくは新たなバックアップサイトを設定する場合，まずはデータベースを同期させる．サービスを一時停止し，スナップショットを取得する．もしサービスを長時間停止させたくなければ，アクセスログを保存しながら，サービスを再開する．スナップショットを新サイトに転送し，スナップショットを復元する．そして，スナップショット取得後のアクセスログを使って，データベースを同期させる．もし，サービス中であれば，データベースの同期完了後に，いったんサービスを停止する．最後に，適切なサイトで AP サーバを起動し，複数の DB サーバに接続し，サービスを再開する．

5.2 不整合回避

アプリケーションからのアクセス要求が DB 接続

表 2 プライマリ DB，バックアップ DB，クライアントの処理状態
Table 2 Status of primary/backup databases and a client.

	プライマリ DB	バックアップ DB	クライアント
ケース 1	処理済み	処理済み	処理済み
ケース 2	処理済み	未処理	処理済み
ケース 3	処理済み	未処理	未処理
ケース 4	処理済み	処理済み	未処理
ケース 5	未処理	処理済み	未処理
ケース 6	未処理	未処理	未処理

ライブラリを介して，各データベースで処理されている間に災害が発生した場合，プライマリ DB，バックアップ DB，クライアントは，**表 2** に示すような処理状態となりうる．

ここで，ケース 2，ケース 4，ケース 5 の場合，バックアップ DB とクライアントとの処理状態に不整合が発生しているため，フェイルオーバ後に問題となる可能性がある．

ケース 4，5 の場合，不整合を解消するには，本来であれば，バックアップ DB においてロールバック作業を行い，当該アクセス要求を未処理の状態に戻す必要がある．しかしながら，一般的なデータベースアプリケーションでは，commit など後戻りできない処理に関して，その応答が返らなくても，処理が正しくできていれば論理的に問題とはならない．クライアント側で，応答がないからといって，複数クリック（処理要求）するようなことがあっても，アプリケーション側で同一処理を行わないような実装を行うためである．通常のアプリケーションにおいては，ロールバック処理は不要である．

ケース 2 では，プライマリ DB とクライアントでは処理済みとなり，バックアップ DB では未処理という不整合，つまり顧客側が予約したにもかかわらず，事業者側がその予約を紛失するようなケースである．

基本的には，バックアップ DB において未処理となっているアクセス要求を処理すればよいが，プライマリサイトが消失した場合，どのアクセス要求がアプリケーションで処理済みか未処理かを判別することは困難である．ただし，バックアップサイトに保存されたアクセスログとバックアップ DB のアクセスログから，バックアップ DB において，どのアクセス要求以後のアクセス要求が未処理かを抽出することは可能である．よって，新しいアプリケーションプログラムをバックアップサイトで起動し，新しいセッションを開始し，未処理のアクセス要求をすべて処理すればよい．アプリケーションが未処理かもしれないアクセス要求までバックアップ DB で処理する可能性はあるが，これはケース 4，5 の状態に相当し，前述したようにこの状態は問題とはならない．

このようにアクセスログを使って，バックアップ DB の処理を進める場合に，問題は 3 つある．1 つは，バックアップサイト側のアクセスログが通信遅延のために，最新状態とはなっていない場合があることである．もし，バックアップサイト側に保存されていないアクセス要求の中に，commit などの確定処理が含まれていなければ，アプリケーションにおいても，当該トランザクションは破棄されるだけなので，問題とならない．もし確定処理が含まれている場合，そのトランザクションは失われ，復旧が困難となる．これに対し，本システムでは，確定処理のアクセス要求に対しては，同期処理を指定することとする．これにより，つねにバックアップサイト，クライアントの順で確定されるので，上記のような問題は生じない．

2 番目の問題は，アクセス応答が異なる，つまりアクセスログ中のアクセス応答内容が，実際のバックアップ DB のアクセス応答内容が異なる場合があることである．両サイトが同期しているならば，当該トランザクションの破棄，ロールバックを行い，アクセス要求前の状態に戻し，アプリケーションにエラーを返せばよい．しかし，両サイトが非同期であれば，アプリケーションに確定処理を返した後にバックアップ DB が異なる応答となる可能性がある．そうなると別途手動で処理するか，バックアップ DB を再構築することで修正するなどの復旧作業を行うが，いずれも手間のかかる作業である．よって，復旧作業が困難なアクセス要求に対しては，同期処理を指定する必要がある．

3 番目の問題は，アクセス要求が異なる，つまりアクセスログ中のアクセス要求内容が，実際のバックアップ DB のアクセス要求内容と異なる場合があることである．たとえば，データベースの内容が異なるような状況で，データベース内のデータを参照した結果を使ってアクセス要求を行うような場合である．このような事態を回避するためには，少なくともデータベースが別途更新されることがないよう処理の順序を管理する必要がある．よって，データベースの状態が更新される可能性があるクエリ実行や更新型のアクセス要求は少なくとも順序確定処理を指定する．

6. 評価実験

6.1 実　　装

提案方式を Java の DB 接続ライブラリを拡張して試作し，耐障害システムに適用して性能評価実験を行った．

JDBC は，Java アプリケーションおよびサーブレッ

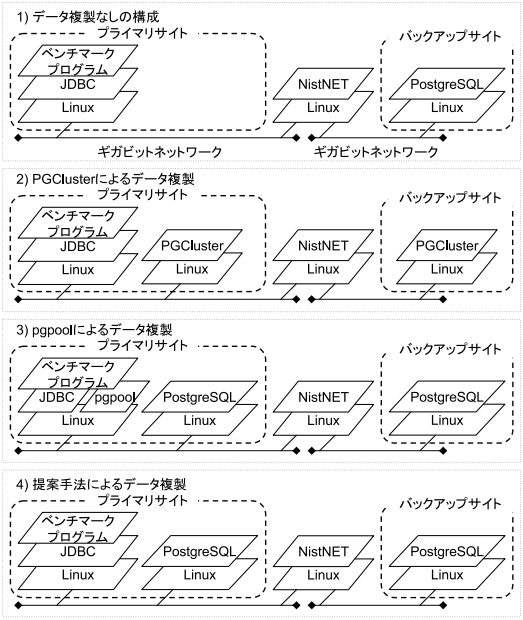

図 4 実験システム
Fig. 4 Experimental system.

トのデータベースアクセスに利用される標準の DB 接続ライブラリである．JDBC は，アプリケーションに標準の API を提供する JDBC ドライバマネージャと，データベース固有のアクセス要求・応答の変換を担う JDBC ドライバから構成される．後者は通常データベースベンダから提供される．我々は，JDBC マネージャの JDBC ドライバを呼び出す部分を拡張した．具体的には，プライマリ DB とバックアップ DB の 2 つの JDBC ドライバをロードし，データベースアクセスを複製し，2 つのデータベースと接続するなどの機能を JDBC ドライバに実装した．このように拡張したライブラリを「JDBC ゲートウェイ（JDBC-GW）」と呼ぶ．

実験では，アクセス状況として，SQL 処理を指定できるようにした．JDBC-GW は，データベースアクセスとアクセス状況（SQL）とを照合し，4 つの複製方式に分類して，処理を進める．具体的には，executeQuery と executeUpdate を順序確定処理とし，commit を同期処理とし，その他の処理を通常アクセス処理とした．

6.2 実験システム

図 4 に実験システムの構成を示す．サーバ OS は Linux（FedraCore5），データベースには PostgreSQL 8.0，および TPC-C [19] に基づく IBM 社 OLTP ベンチマークツールキットを用いた．インターネットでの利用を想定し，遅延およびパケットロスをエミュレートするために NistNET [2] を用いた．実験では，NistNET のパラメータを変動させ，さまざまなネットワーク環境下での性能検証を行い，以下の処理構成の性能比較を行った．1) データ複製なしの構成，2) データベース内部でのデータ複製を想定し，PGCluster [18] によるデータ複製を行う構成☆，3) データベース外部でのデータ複製を想定し，pgpool [23] によるデータ複製を行う構成，4) 提案方式 JDBC-GW によるデータ複製を行う構成．

Pgpool のシステム構成では，プロキシサーバをアプリケーションサーバ内で動作させ，ベンチマークプログラムはローカルなソケット通信を介して接続した．

6.3 実験結果

実験結果を**図 5** に示す．実験では，NistNET を用いて，RTT のネットワーク遅延を 0 ミリ秒，1 ミリ秒，10 ミリ秒に設定し，またパケットロス率を 0%，0.1%，および 1%に設定した．グラフの縦軸は，データベースのスループットを毎秒のトランザクション数（tps）で示す．データ複製なしのローカルなデータベースアクセスでは，11.4 tps である．データ複製なしの遠隔データベースアクセスでは，ネットワーク遅延とパケットロスの大きい影響を受け，処理性能が低下している．トランザクションの中で，多くのファイルシステムアクセスがあり，それらがボトルネックとなっていると思われる．

PGCluster と pgpool の場合，データ複製時の性能は，データ複製なしのローカルなデータベースアクセスの場合と比較して，55.6%から 92.5%下落した．これに対して，提案方式の JDBC-GW の場合，性能の劣化は 32.6%から 60.6%にとどまっており，特に，ネットワーク遅延・パケットロスに対する許容性が高いことが分かる．データベースアクセスに着目すると，提案方式では，ローカルなデータベースアクセスのみ，非同期遠隔アクセス，同期遠隔アクセスがあるが，ネットワーク遅延・パケットロスが大きくなると，同期遠隔アクセスの影響のみが目立つ．

次に，障害発生時のシステムの動作検証では，上述の TPC-C プログラム中で，同期複製の問題となる commit 処理に関して，処理前，処理中，処理後に強制終了のコードを挿入し，動作させた．フェイルオーバ後に，正常な TPC-C プログラムを実行し，バックアップサイトで動作させ，処理数が正常であることを確認した．

☆ PostgreSQL 7.4.7 ベース

複製方式	遅延(ms)	パケットロス(%)	Tx/sec	性能劣化率	複製方式	遅延(ms)	パケットロス(%)	Tx/sec	性能劣化率
データ複製なし	0	0	11.42	0.0%	pgpool	0	0	5.07	55.6%
		0.1	10.51	8.0%			0.1	4.29	62.4%
		1	4.78	58.1%			1	1.67	85.4%
	1	0	8.66	24.2%		1	0	4.01	64.9%
		0.1	8.08	29.2%			0.1	3.68	67.8%
		1	4.31	62.3%			1	1.3	88.6%
	10	0	2.85	75.0%		10	0	1.33	88.4%
		0.1	2.76	75.8%			0.1	1.35	88.2%
		1	2.16	81.1%			1	0.86	92.5%
PGCluster	0	0	4.1	64.1%	提案方式	0	0	7.7	32.6%
		0.1	3.64	68.1%			0.1	6.9	39.6%
		1	2.5	78.1%			1	5	56.2%
	1	0	3.62	68.3%		1	0	6.9	39.6%
		0.1	3.48	69.5%			0.1	6.6	42.2%
		1	2.04	82.1%			1	4.5	60.6%
	10	0	1.8	84.2%		10	0	6.5	43.1%
		0.1	1.72	84.9%			0.1	6.2	45.7%
		1	1.45	87.3%			1	5.3	53.6%

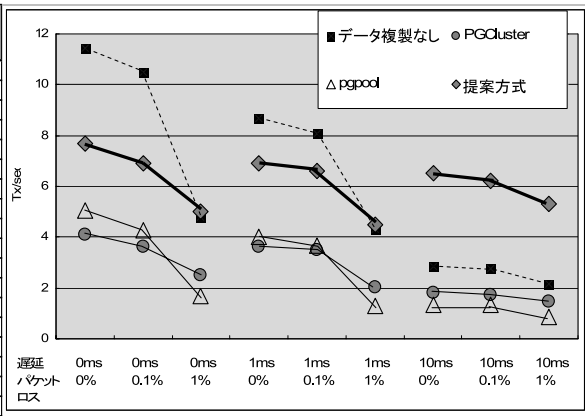

図 5 実験結果
Fig. 5 Experimental results.

7. 関連研究

既存の関連技術について，処理レイヤごとに述べる．ストレージレイヤでのデータ複製に関しては，RAID[13] がよく知られている．RAID が 1 つの計算機システムに適用される技術に対し，ネットワーク上のシステムに拡張された RADD と呼ばれる機構が提案されている[17]．RADD は，コンピュータ間でデータを保護する基本的な耐障害システムといえるが，LAN のみを対象としており，広域ネットワーク上での動作性能は検証されていない．

また，最近 iSCSI[4] を利用したデータ複製の開発が活発である．これは，専用線よりもインターネットを利用することで，コスト削減を進めたいなどの市場ニーズが要因の 1 つにある．小サイズのデータをプライマリサイトとバックアップサイトとで高頻度にやりとりするため，すべてのサーバが集積され，高性能ネットワークに接続されているようなデータセンタでは有効かもしれないが，遠距離間では実用的な性能を期待することはできない．

ネットワークレイヤでのデータ複製では，NFS サーバ/クライアント間の通信データを監視し，バックアップサーバに転送し，複製を行う手法[22] などがある．この手法では，通信データの監視サーバおよびバックアップサーバのネットワークと既存システムとを切り離すことができ，システム性能に影響を与えることなく複製処理を実現できる．しかしながら，バックアップサーバの処理状態を既存システム側にフィードバックすることが困難であり，整合性を保持することは難しい．さらに，パケットロスへの対応も困難である．

ファイルシステム（OS）レイヤのデータ複製に関しては，Spiralog[10] と ReiserFS[12] などで使われているファイルシステムログ技術が関連する．これらのシステムでは，ファイルアクセスログが保護され，システム障害のためにファイルが壊れた場合に，そのファイルの復旧に利用される．最新のファイルシステム（OS）の多くは，スナップショット機能を有し，任意の時点のファイルシステム状態を保存，復元することが可能となってきている．これらのファイルシステム技術は，ファイルの復旧には十分ではあるが，アプリケーションやデータベースの状態を復旧するには，ファイルだけでなく，処理状態も含め，統合的に管理，制御する機能が不足している．本提案は，ファイルシステム技術を置き換えるものではなく，このような統合管理機能を補完するものである．

データベースレイヤのデータ複製に関しては，分散データベースの領域で多くの研究がなされてきている．データベース複製技術[1),18)] は，可用性，耐障害性を向上させるが，分散データベース間での安定したネットワークを前提としている．そのため，インターネット上での広域で利用するようには設計されていない．一方，グリッド技術は，広域分散環境を想定した技術である．DPSS[5] や CMS データグリッド[8] では，高速かつ高信頼のデータ複製を実現してきている．しかし，信頼性要件（RTO，RPO）に適応するような機能は考慮されていない．

8. おわりに

本論文では，柔軟な障害復旧機構を提案した．汎用的な DB 接続ライブラリを拡張し，あらかじめ登録されたデータベースアクセス状況と，実状況とを照合し，データ複製方式を制御することで，柔軟性を実現する．

アクセス状況の登録を変更することで，状況と処理：通常アクセス（非同期処理），同期処理，順序確定処理，スキップ処理との対応付けを変更し，IT サービスの災害時復旧要件 RTO，RPO に応じての設定を行うことができる．

実験では，TPC-C ベンチマーク（一般的な卸売り会社のトランザクション処理）を用いて，従来手法と性能比較を行った．データベースの同期更新頻度を調整することで，提案手法はネットワーク遅延とパケットロスの両方に従来手法と比べ許容性があることが確認できた．東京〜大阪でディザスタリカバリ目的で同期複製を行おうとすれば，およそ遅延は 10 ミリ秒であり，その環境では，既存手法の 3.6 倍（提案手法：6.5 tps，PGCluster：1.8 tps）優れている．中小規模でのコマースサイトの更新性能要件は 5〜10 tps といわれており，本実験では限定的ながら，提案方式が実用可能レベルであることを確認できた．

今回の実験では，AP サーバが 1 つの構成であり，順序制御に関しては動作検証をしておらず，今後の課題となる．また，SQL のアクセスログなどから，ボトルネックを分析し，同期や順序確定の複製処理を最適化するなどの性能向上の検討を進める予定である．

参 考 文 献

1) Carey, M.J. and Livny, M.: Distributed Concurrency Control Performance: A Study of Algorithms, Distribution, and Replication, *Proc. 14th VLDB Conf.*, pp.13–25 (1988).
2) Carson, M. and Santay, D.: NIST Net: A Linux-based Network Emulation Tool, *ACM SIGCOMM Computer Communications Review*, Vol.33, No.3, pp.111–126 (2003).
3) Cecchet, E., Marguerite, J. and Zwaenepoel, W.: C-JDBC: Flexible Database Clustering Middleware, *Proc. USENIX Annual Technical Conference, Freenix track* (2004).
4) Chang, F., Ji, M., Leung, S.-T.A., MacCormick, J., Perl, S.E. and Zhang, L.: Myriad: Cost-effective Disaster Tolerance, *Proc. USENIX Conference on File and Storage Technologies*, pp.103–116 (2002).
5) Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S.: The data grid: Towards architecture for the distributed management and analysis of large scientific datasets, *Journal of Network and Computer Applications*, Vol.23, No.3, pp.187–200 (2000).
6) Geiger, K.: *Inside ODBC*, アスキー (1996).
7) Gray, J., Helland, P., O'Neil, P. and Shasha, D.: The Dangers of Replication and a Solution, *Proc. 1996 ACM SIGMOD*, Vol.25, No.2, pp.173–182 (1996).
8) Holtman, K.: CMS Data Grid System Overview and Requirements, *The Compact Muon Solenoid Experiment Note 2001/037* (2001).
9) ISO/IEC10026-1:1992, Information technology —Open Systems Interconnection—Distributed Transaction Processing—Part 1: OSI TP Model (1992).
10) Johnson, J.E. and Laing, W.A.: Overview of the Spiralog File System, *Digital Technical Journal of Digital Equipment Corporation*, Vol.8, No.2, pp.5–14 (1996).
11) Kemme, B. and Alonso, G.: Don't be lazy, be consistent: a new way to implement Database Replication, *Proc. 26th International Conference on Very Large Databases*, Vol.26, pp.134–143 (2000).
12) MacDonald, J. and Reiser, H.: Reiser4 Transaction Design Document, Technical report, Namesys (2001).
13) Patterson, D.A., Gibson, G. and Katz, R.H.: A case for redundant arrays of inexpensive disks (RAID), *Proc. 1988 ACM SIGMOD*, pp.109–116 (1988).
14) Reese, G.: JDBC による Java データベースプログラミング，オライリー・ジャパン (2001).
15) Ronstrom, M. and Thalmann, L.: MySQL Cluster Architecture Overview, *MySQL Technical White Paper* (2004).
16) Sceppa, D.：プログラミング Microsoft ADO.NET，日経 BP ソフトプレス (2002).
17) Stonebraker, M. and Schloss, G.A.: Distributed RAID—A New Multiple Copy Algorithm, *Proc. 6th IDEC*, pp.430–437 (1990).
18) Tanida, Y. and Mitani, A.: PGCluster (2005). http://pgcluster.projects.postgresql.org/
19) TPC: Standard Secification Revision 5.2, Technical report, Transaction Processing Performance Council (TPC) benchmark C (2003).
20) Wolfson, O. and Milo, A.: The multicast policy and its relationship to replicated data placement, *ACM Trans. Database Syst.*, Vol.16, No.1, pp.181–205 (1991).
21) 小野田英樹，波多野賢治，宮崎 純，植村俊亮：ウェアラブルコンピューティングのための追記型ファイルシステムの実装，第 15 回データ工学ワークショップ予稿集，Vol.DEWS2004, 1-A-02 (2004).
22) 種村昌之，新城 靖，板野肯三，千葉 滋：ネットワークの監視によるバックアップシステムの実現，情報処理学会シンポジウム論文集，Vol.2001, No.16, pp.57–64 (2001).

23) 石井達夫：pgpool: Connection Pool Server for PostgreSQL (2006).
http://pgpool.projects.postgresql.org/

(平成 18 年 7 月 7 日受付)
(平成 19 年 1 月 9 日採録)

中村　暢達（正会員）

1989 年東京大学工学系研究科精密機械工学科専門課程卒業．1991 年同大学院同研究科精密機械工学専攻修士課程修了．同年日本電気（株）入社．マルチメディア通信，ユーザインタフェース，コンテンツ保護，耐障害システム，ユビキタスシステムに関する研究開発に従事．現在，奈良先端科学技術大学院大学情報科学研究科博士後期課程在籍．

藤山健一郎（正会員）

1999 年筑波大学第三学群情報学類卒業．2001 年同大学大学院工学研究科情報科学修士号取得後中退．同年日本電気（株）入社．現在，同社インターネットシステム研究所にて，主に耐障害性，高可用性に関する研究に従事．

河合　栄治（正会員）

1996 年京都大学理学部数学科卒業．1998 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．2001 年同大学同研究科博士後期課程修了．2000 年科学技術振興事業団研究員．2003 年奈良先端科学技術大学院大学附属図書館研究開発室助手．2006 年より奈良先端科学技術大学院大学情報科学研究科特任助教授．高速ネットワークシステム，大規模情報配信システム，システムセキュリティ技術の研究に従事．博士（工学）．

砂原　秀樹（正会員）

1983 年慶應義塾大学工学部電気工学科卒業．1988 年同大学大学院博士課程修了．同年電気通信大学情報学部助手．1994 年奈良先端科学技術大学院大学情報科学センター助教授．2001 年同大学情報科学センター教授．2005 年同大学情報科学研究科教授，現在に至る．工学博士．インターネット，大規模広域分散環境，ネットワーク，並列処理，オペレーティングシステム，電子図書館に関する研究に従事．ACM，IEEE 各会員．