1K-5

データベース同期複製のための JavaAPI の拡張

藤山　健一郎　　中村　暢達　　平池　龍一
ＮＥＣ　インターネットシステム研究所

1. はじめに

従来、障害によるデータの破壊や損失を防ぐ手法として、データベース（DB）のレプリケーション機能を用いてデータを複製する方法がある。レプリケーション[1]とは、複製元であるプライマリ DB が行ったデータ処理のログ等を、複製先のバックアップ DB に転送し、同じデータ処理を行わせることで、DB ごとデータの複製を行う技術である。

データ処理毎にプライマリ DB とバックアップ DB で同期を取るレプリケーションを Eager レプリケーションと呼ぶ。データの損失を最小限にするには、この Eagar レプリケーションが必要となるが、Eagar レプリケーションは商用 DB の高価な上位製品のみでしか提供されていない。一部のフリーDB には、Eager レプリケーションを付加する拡張ソフトも存在するが、その DB 専用のものであり、かつ AP のソースを書き換える等の作業が必要なため、導入は困難である。したがって、Eager レプリケーションを備えない DB で構成されたシステムの場合、DB を移行する必要があるが、データ管理の要である DB の移行には大きなリスクが伴う。しかも、２系統分の DB が必要なため、商用 DB に移行する場合は、構築コストが大幅に増加してしまう。

本稿では、既存システムで用いられている DB の種類にかかわらず汎用的に、かつ、そのシステムの構成を変えることなく容易にレプリケーション機能をアドオンし、DB 同期複製を実現する方式を提案する。

2. JavaAPI 拡張による DB 同期複製

2.1. 方式提案

DB と連携したアプリケーション（AP）では、直接 DB にアクセスするのではなく、汎用的な DB 接続を提供する API を介してアクセスするのが一般的である。図 1に接続 API を介して DB アクセスを行うシステムのデータ処理の流れを示す。接続 API は、AP、DB の両方から独立した共通基盤であり、AP が発行した DB 非依存なデータ処理要求を、個々の DB 固有のものに変換することで、DB の種類によらない汎用的な DB アクセスを可能にする。

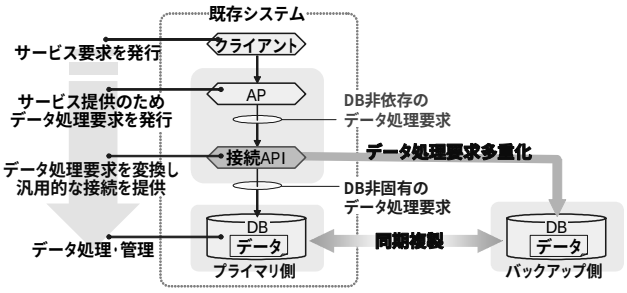

図 1：接続 API を介した DB 接続と提案方式

以上に述べた特徴を鑑み、接続 API におけるデータ処理要求を多重化することで DB 同期複製を行う方式を提案する。すなわち、AP が接続 API を介して発行しようとする DB 非依存なデータ処理要求をトラップし、複製した後、改めて接続 API を介して、プライマリ DB、バックアップ DB それぞれにデータ処理要求を発行する。このようにすることで、両方の DB で同じデータ処理を行わせて、DB 同期複製を実現する。

前述のとおり、接続 API は独立した共通基盤であるため、既に接続 API を介して DB アクセスを行っているシステムであれば、どのような DB が用いられていても、既存システムの構成を変えることなく、提案方式を適用することができる。

2.2. JavaAPI への実装

Java における接続 API である JDBC を拡張し、提案方式を実装した。JDBC にデータ処理要求を多重化する機能を組み込み、JDBC へのデータ処理要求をトラップ、複製してから、本来の JDBC に転送するようにする。この拡張機構を JDBC ゲートウェイ（JDBC-GW）と呼ぶ。JDBC-GW を用いた場合の AP から DB へのデータ処理要求の流れを図 2に示す。

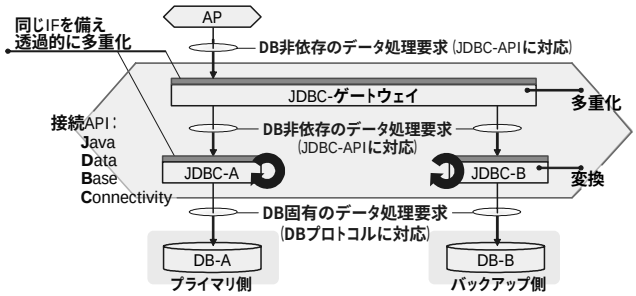

図 2：JDBC-GW を介した DB 接続

なお、JDBC-GW は JDBC と同じインタフェースを備えることで、AP からは普通の JDBC として、JDBC からは AP として認識される。つまり、既存システムに対し影響を与えず、透過的に扱われるため、

Enhanced Java-API for Synchronous Database Replication.
Ken-ichiro FUJIYAMA, Nobutatsu NAKAMURA,
and Ryuichi HIRAIKE.
Internet Systems Research Laboratories, NEC Corporation.

システムの構成を変更する必要がない。

2.3. 動作

通常時の動作（DB 同期複製）

JDBC-GW が、AP の発行するデータ処理要求を多重化することで、DB 同期複製を行う。両 DB からのデータ処理の応答は、一旦、JDBC-GW が受けて、プライマリ DB の応答のみを AP に返す。AP にとっては、通常通り JDBC を介してデータ処理要求を発行して応答が１つだけ返ってくるので、複製処理は意識されない。

障害発生時の動作（データ保護）

プライマリ DB に障害が発生した場合、プライマリ DB のデータが破壊されても、バックアップ DB には破壊前のデータが複製されている。また、データが破壊されなくても、データ更新中に障害が発生した場合は、プライマリ DB はデータ処理が中途状態となり不整合が発生する可能性がある。しかし、バックアップ側は、データ処理要求の段階で多重化されているため、プライマリ DB での障害の影響を受けず、正常にデータ処理を行えるので、不整合は発生しない。このため、バックアップ DB の整合の取れたデータを用いて、データの損失なくサービスを継続、すなわち RPO（Recovery Point Objective）＝0 でサービスを復旧できる。

障害発生後の動作（フェイルオーバー）

JDBC-GW は障害の発生したプライマリ DB を切り離し、稼動しているバックアップ DB にのみデータ処理要求を発行する。バックアップ DB には整合のとれたデータが保存されているため、正常なデータ処理を行い、応答を返す。バックアップ DB の応答は JDBC-GW を介して AP に返されるため、AP にはそれがどの DB の応答かは意識されない。そのためプライマリ DB の障害は隠蔽され、無停止、すなわち RTO(Recovery Time Objective)＝0 でサービスを復旧できる。

以上のように、障害時にデータの損失なく、無停止でサービスを継続することができる。

3. 評価実験

JDBC-GW による DB 同期複製機構を、実際に販売、運用されている WEB アプリケーション製品に適用し、同期複製のオーバーヘッドにより、どの程度性能が低下するかを検証する評価実験を行った。

3.1. 実験環境

実験システムの構成を図 3 に示す。AP サーバマシンでは、AP サーバとして BEA WebLogic Server 8.1、WEB アプリケーション製品として住宅業向け工事発注 ASP サービス（以下、発注 ASP）を使用した。２つの DB サーバマシンでは、それぞれ DB として Oracle9i Database Standard Edition を使用した。クライアントマシンには、発注 ASP にサービス要求を発行して、処理時間を計測する実験ツールとして、Microsoft Web Application Stress Tool 1.1（以下 WAS ツール）を使用した。

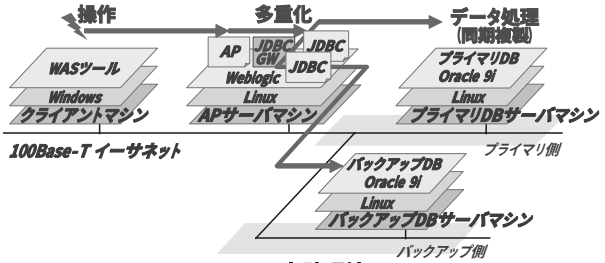

図 3：実験環境

3.2. 実験方法・結果

提案方式を適用していない発注 ASP と、適用した発注 ASP それぞれに、WAS ツールを用いて、発注作業を行う一連のサービス要求を発行し、各要求の応答時間を計測した。

No.	サービス要求種類	応答時間[ms] 未適用時	適用時	差分[ms]	比率[%]
1	開始	142.9	166.4	23.5	16.4
2	ログイン1	184.9	224.8	39.9	21.6
3	ログイン2	163.6	152.3	-11.3	-6.9
4	ログイン3	90.5	138.2	47.7	52.7
5	ログイン4	155.8	166.6	10.8	6.9
6	発注開始	1058.8	1463.1	404.3	38.2
7	登録	508	641.7	133.7	26.3
8	確認1	145.3	207.9	62.6	43.1
9	確認2	113.3	210.3	97.0	85.6
10	確認3	119	170.1	51.1	42.9
11	確認4	132.2	153.7	21.5	16.3
12	確定1	538.8	944.6	405.8	75.3
13	確定2	616	829.3	213.3	34.6
14	終了	105.6	158.6	53.0	50.2
	合計	4074.7	5627.6	1552.9	+38.1

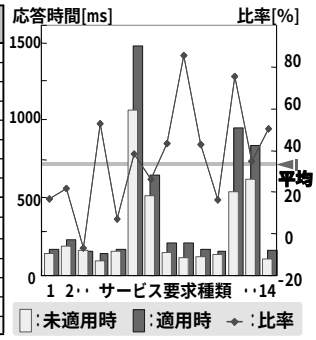

図 4：実験結果

実験結果を図 4に示す。作業全体で応答時間が約 38%延びている。なお、１サービス要求あたりの応答時間の遅れは平均 110ms 程度である。発注 ASP のような WEB アプリケーションでは、ユーザによる待ち時間等が発生するため、それに比べれば、提案方式を適用したことによる応答時間の遅れは十分小さく、実用上は問題ないと考える。実際に、手動で実験システムを操作したところ、体感的なレスポンスの低下は感じられなかった。

4. おわりに

本稿では、単一 DB からなる既存システムに、DB 同期複製機能を容易にアドオンする、API 拡張による DB 同期方式について述べた。提案方式では接続 API を拡張することで、DB の種類によらず、また既存システムの構成を変更することなく、DB 複製機能を実現できる。また本方式を Java における接続 API である JDBC に実装し、実運用製品に適用し評価を行った。その結果、実用的な性能が得られることを確認した。

今後は複数 AP 間の順序制御機能、災害対策を考慮した遠距離複製機能等の開発を行う予定である。

参考文献

[1] J.Gray et al, "The Dangers of Replication and a Solution" ACM SIGMOD Conference, 1996.