データベースレプリケーションの実装によるLinuxサーバの二重化構成

Configuration of Duplex Linux Server by Implementing Database Replication

中島　龍太朗　　大和　浩之
Ryutaro Nakashima　Hiroyuki Yamato

概要

近年のITの普及により、多くの企業が、様々なサービスを提供するために情報システムを利用している。これに伴い、システムの重要度は年々高まり、同時にシステム停止がもたらす損害や社会への影響も大きくなってきている。このような中、システムを停止させないための様々な技術が注目を集め、システムを構築する上でも重要な要素となってきている。

このひとつとしてサーバなどを二重化する方法があるが、二重化によるコストの増大は避けることができない。そこで、コストの削減策として、安価に高性能なサーバを構築できるLinuxを採用することが考えられる。Linuxに関しては、日本国内でも、基幹業務システムに採用する事例やデジタル家電に組み込むなどの事例が増えているが、Linuxサーバの稼動実績はまだ多くはない。

本稿では、サーバのOSにLinuxを採用したグローリー工業様の遊技市場向け会員管理システムを事例とし、システムの信頼性向上のために搭載したフォールトトレラント機能の実装に関して述べる。特に、この中でも重要な二重化したサーバのデータベースの同期をとるためのレプリケーション処理と、障害発生時および障害からの復旧時に行う処理について、構想段階から最終的な実装までに発生した課題と、その対策方法について述べる。

1. はじめに

1.1 グローリー工業様について

グローリー工業様（以下、お客さま）は、金融・流通・レジャー場向けに、入出金機・両替機などの貨幣処理機や自動販売機などの機器を国内外に製造・販売しているメーカーである。特に、貨幣処理機に関してはパイオニア的存在であり、業界のリーディングカンパニーである。

本稿では、お客さまが遊技市場向けに製造している会員管理システムであるP-BANK SQUARE V（以下、本システム）をLinuxサーバ二重化の事例として紹介する。

1.2 P-BANK SQUARE V システム概要

本システムは、クライアント/サーバ型のシステムであり、パチンコホール（以下、ホール）に設置され、おもに来店した顧客が利用・操作する機器（以下、端末）、それら端末から情報（データ）を収集し蓄積するサーバ、情報の参照や各種の設定を行うための分析クライアント（パソコン）で構成される。本システムの主な機能として、会員情報の管理・売上情報の管理・景品情報の管理などがある。なお、本稿で取り上げているサーバには、システム商品として低価格で数多く出荷することを可能にするため、1台当たりの製造コストを下げることができるLinuxをOSに採用し、データベースにInterBaseを採用している。

これらの構成を図1に示し、それぞれの役割を説明する。

図1下段の端末では、上段左のサーバとデータを送受信することにより、様々なサービスを実現している。具体的な機能の一部として、会員から玉やメダルを預かる（貯玉）サービスや

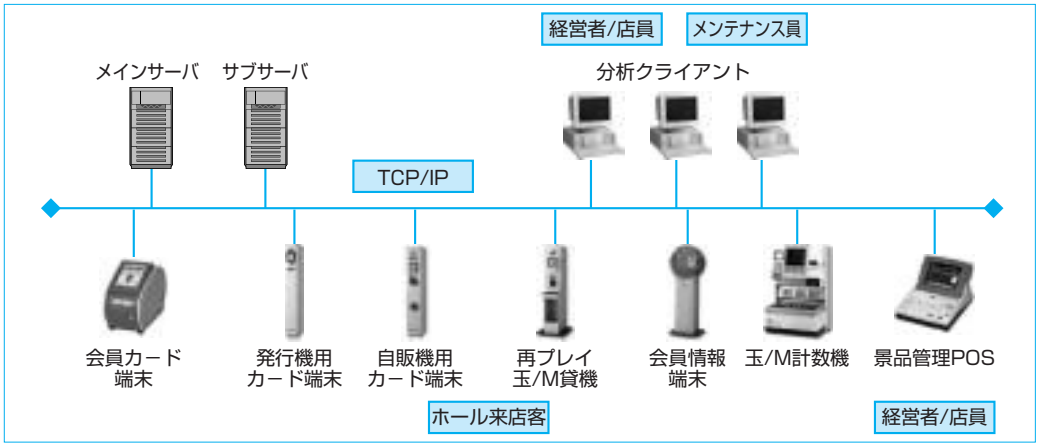

図1　システム全体の構成図

貯玉を払い出す（再プレイ）サービスなどがある。また、上段右の分析クライアントでは、端末と同様、サーバとデータを送受信することにより、ホールの運営に関する設定や売上情報の参照や帳票出力などのサービスを実現している。この分析クライアントの利用者はホールの従業員もしくは販売店などから派遣されるメンテナンス員である。

2. サーバの二重化構成

2.1 システム要件

前述したように本システムのサーバは、ホール内に設置された端末と通信することで利用する顧客に様々なサービスを提供しており、システム全体の中心である。また、これらのサービスの中には、ホールの営業に欠かせないものも多く、ハードウェア等の障害が発生したとしても、サーバの機能を停止させないしくみが必要である。本システムでは、サーバを二重化することにより対応することとした。以下は、サーバを二重化するにあたり、指針となったシステム要件である。

① 二重化した（複数の）サーバ間でデータの同期をとること
② 片方のサーバのみでの運用（単体運用）を可能にすること
③ 復旧処理により単体運用時のデータを復元できること
④ Linuxやデータベースの知識がなくても操作できること

まず①については、障害発生直前までのデータを復旧するために必要な要件であり、データの更新処理が、両方のサーバに対してリアルタイムに反映されることが求められた。②については、障害発生時に、可能な限り営業を停止させないための要件であり、障害が発生したサーバを切り離し（サーバ切替え処理）、サーバ1台のみでの運用（単体運用）が可能になるよう求められた。③については、単体運用から二重化の運用形態に復帰させるための要件であり、障害が発生したサーバを交換することにより、二重化の運用形態に復帰できることが求められた。④については、これまでの3つの要件すべてに関わることである。本システムの運用の中心は、LinuxやInterBaseに関して知識をもたないホールの従業員やメンテナンス員であるため、障害発生時に運用切替えを行う場合、可能な限りLinuxおよびInterBaseを意識させないことが求められた。次節より、これらの要件を満たすサーバ構成について検討した内容を述べる。

2.2 二重化の実装方法

上記の要件の中で最も重要なことであるサーバ間でデータの同期を取る手法として、次の2つの実装方法について検討した（表1）。

① InterBase用レプリケーションサーバの利用
② 独自のレプリケーション管理の開発

まず、①はInterBaseのデータレプリケーションシステムであるIBReplicatorを使用する方法である。この実装方法を採用した場合の利点としては、データの更新処理が他方のサーバに対して、データベースエンジンやドライバなどを経由せずに直接行われるため高速な動作が期待できること、InterBase独自の拡張機能の利用やレプリケーションを行うタイミングなど、InterBaseの機能をフルに使用した動作設定が可能であることが挙げられる。しかし、いくつかの調査をした結果、レプリケーション処理のエラー状況の監視や運用切替え時にレプリケーションサーバを制御することは、困難であると予測された。

一方、②の独自にレプリケーション管理を開発する方法では、

障害検知や障害発生時の対応が柔軟にできる。サーバ切替え・復旧処理に関して、ハードウェアやネットワークの障害によりレプリケーション処理が正常に動作しなかった場合の仕様をシステムの運用に合わせて実装することができるなど、きめ細かな仕様を盛り込めることが予測された。さらに、単体運用時のデータのバックアップ機能や処理の自動化なども搭載が可能となるなど、機能拡張の視点からみても優れていることが予測できた。

表1　レプリケーション処理の実装方法の長所と短所

	長　所	短　所
IBReplicator	•高速な動作 •InterBase独自の機能を利用可能 •高度な動作設定が可能	•単体運用でのバックアップ方法が別に必要 •エラー処理を規定できない •復旧処理などでサービスの起動・停止が必要
独自開発	•復旧処理などでの制御が容易 •単体運用でのバックアップ機能を搭載可能 •エラー処理を規定できる •機能追加など柔軟に対応可能	•開発コストが必要 •IBReplicatorに比べ低速な動作

このような点と、独自開発のために必要となるコストに関して、本システムが数多く出荷されるシステム製品であることからクリアできるという前提から、独自のレプリケーション管理を実装する方法を採用した。

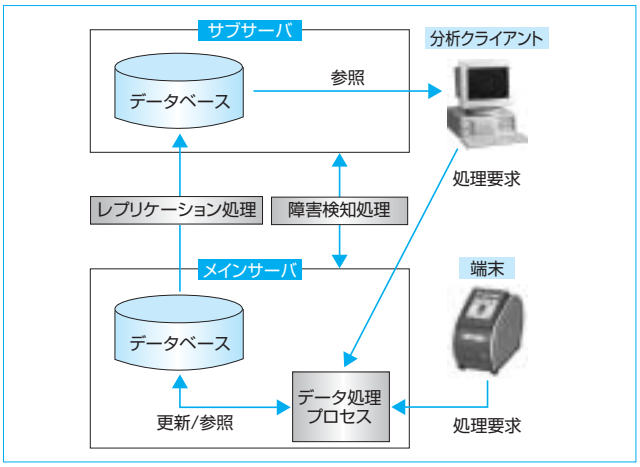

図2　サーバ二重化の構成

二重化したシステムの構成を図2に示す。二重化したサーバのそれぞれを、メインサーバ・サブサーバと呼んでいる。分析クライアントや端末からのデータ更新処理はメインサーバに対して行うしくみになっており、更新されたデータはレプリケーション処理によりリアルタイムにサブサーバに反映する。また、メインサーバの負荷を軽減するため、分析クライアントからのデータベースの参照はサブサーバに対して行っている。その他、システムの障害を検知するため、メインサーバとサブサーバはお互いの稼動状態を監視している。

次に、障害発生時の運用切替えについて述べる（図3）。障害が発生した場合、一方のサーバをシステムから切り離し、サーバ1台のみでシステムの稼動を行う。この際、障害が発生したサーバに割り振られていたIPアドレスを、稼動しているサーバにエイリアスIPアドレスとして割り当てる。エイリアスIPアドレスは、ひとつのNIC（Network Interface Card）に対して複数のIPアドレスを仮想的に設定する機能（IPエイリアス）を用いて割り当てられたIPアドレスである。これにより、障害の発生したサーバのIPアドレスを他のサーバに割り当てることにより、分析クライアントや端末からはどちらのサーバが稼動しているかを意識する必要がなくなる。

一方、故障したサーバを交換したときには、IPアドレスの割り当てを元に戻し、稼動していたサーバから交換したサーバにデータを取得することで通常の二重化システムに復旧することを可能とした。

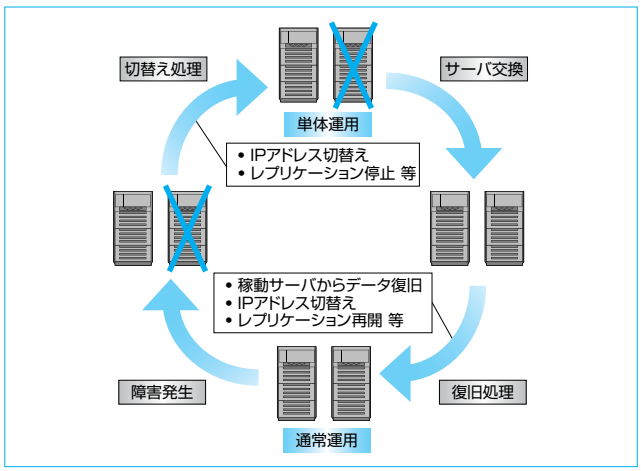

図3　障害・復旧時のサーバ状態の遷移

3. レプリケーション管理

3.1 サーバ内部構成と処理概要

サーバの内部構成と処理概要を図4に示す。主要なプロセスとして端末・分析クライアントなどからの通信コマンドを受け付ける通信受諾プロセス、各種データの更新を行うデータ更新プロセス、レプリケーションを行うレプリケーションプロセスがある。また、メインサーバとサブサーバ間で双方の状態を確

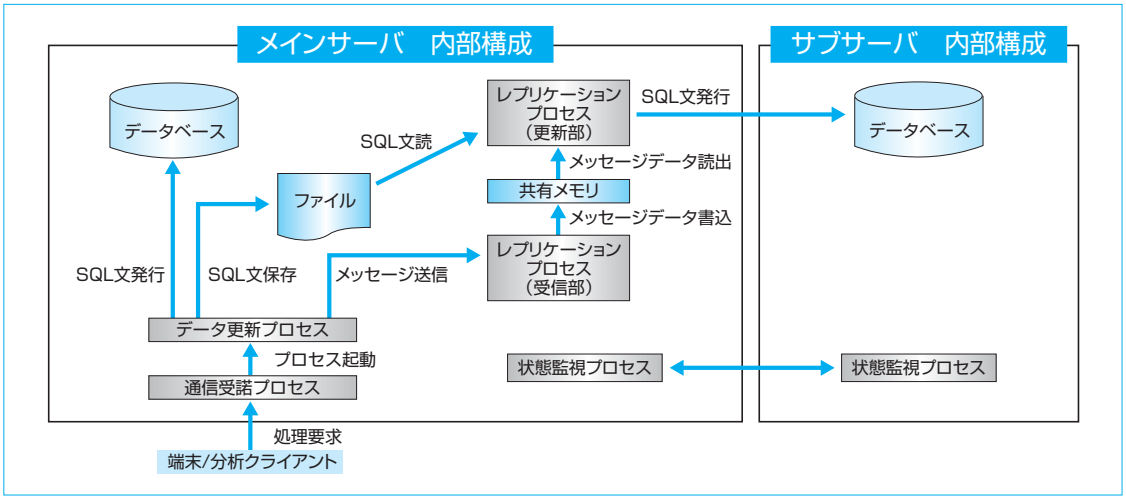

図4　レプリケーション処理のサーバ内部構成

認するための状態監視プロセスも存在している。

これらのプロセスはそれぞれ独立しており、プロセス間のデータの受け渡しはプロセス間通信（IPC：InterProcess Communication）により実現している。データの更新プロセスとレプリケーションプロセスの間では、メッセージキューを用いるが、メッセージキューの数とサイズはシステム資源としてLinuxのカーネルパラメータであらかじめ設定されているために、これらの上限に到達しないように注意する必要がある。

メインサーバ上でデータベースを更新したデータ更新プロセスは、更新時に使用したSQL文をファイル形式で保存し、コミットされたタイミングでメッセージをレプリケーションプロセスに送信する。レプリケーションプロセスは、更新メッセージを受け取る受信部と実際にサブサーバのデータベースを更新する更新部の2つのプロセスに分割されている。受信部では、メッセージの受け取りとメッセージデータの共有メモリへの書き込みを行う。共有メモリへの書き込み後、このプロセスは、次のメッセージを受け取るために受信待ち状態となる。一方、更新部は定期的に共有メモリの内容を読み込み、これが更新されたことをトリガーとしてデータ更新プロセスが作成したSQL文を読み込む。その後、読み込んだSQL文を使用してサブサーバのデータベースを更新している。

以上の構成でレプリケーション処理を実装したことにより、次のような効果が得られた。

① プロセス間通信を用いてデータの更新を非同期に実行することにより、複数のデータ更新処理のレプリケーション部分をひとつに集約することができ、アプリケーションを容易に作成できる

② レプリケーションプロセスのプログラム構成をメッセージ受信部とデータベース更新部に分割することにより、数やサイズに制限のあるメッセージキューを有効に利用することができる

しかし、データ更新処理とレプリケーション処理を独立させたことにより、サーバ間でのデータの整合性が損なわれる問題、および、双方のサーバのデータが一致するまでにタイムラグが発生するという問題が生じた。これらの問題への対策については次節以降で述べる。

3.2 データ整合性の確保

データ更新プロセスとレプリケーションプロセスを非同期に動作させることにより、レプリケーションプロセスで障害が発生した場合、サーバ間でのデータの不一致が発生する。そのため、障害発生後のデータの整合性を確保するために次のような対策を行った。

（1）行単位での更新（図5）

データベースを更新するときは、行の全項目を更新するSQL文を発行するようにプログラムを作成した。必要な項目だけを更新するSQL文を用いた場合、障害の発生により更新されなかった項目は、同じ行の同じ項目が更新されない限りサブサーバのデータベースに反映されない。一方、行単位で全項目を更新するように作成すると、障害が発生しても次に同じ行が更新されるとサブサーバのデータベースに反映され、データの整合性が回復される。

（2）データベースの複製

レプリケーション処理のみでは、エラーが長時間にわ

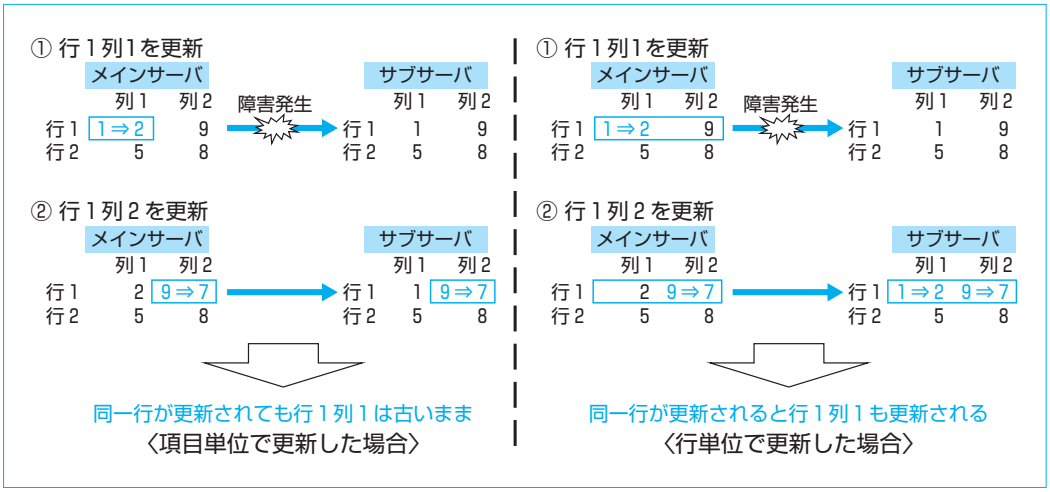

図5　障害発生時における更新結果の違い

たって複数累積するとサーバ間でデータの不整合が進行する可能性がある。したがって、万が一を想定し、原則として1日に1回、メインサーバからサブサーバにデータベースの複製を作成し、両者のデータを完全に一致させるというしくみも取り入れている。

3.3 タイムラグの軽減

データ更新プロセスがメインサーバのデータベースを更新した後、この更新内容がサブサーバへ反映されるまで双方のサーバではデータが一致しない。このタイムラグを短縮するためにレプリケーション処理のパフォーマンスをできる限り向上させる必要があった。そのため、データベースの更新に関する以下の部分について、パフォーマンスを調査した。

① データベースとの接続
② トランザクションの開始
③ SQL文の実行
④ トランザクションのコミット
⑤ データベースとの接続解除

それぞれに要する時間をログ採取などで計測した結果、「① データベースとの接続」がレプリケーション処理のレスポンスに大きな影響を与えることがわかった。特に、ネットワークを介した接続の場合に、この傾向は顕著に現れていた。この対策として、データベースとの接続回数を減少させ、レプリケーション処理のパフォーマンスの向上を図った。具体的には、データベースとの接続を解除するタイミングを見直し、一定期間以上の更新が行われなかった場合に解除するようにした。その結果、データを連続して更新したときにレスポンスが向上した。一方、接続を維持することにより発生した課題もあった。例えば、ネットワークの切断などの予期せぬ現象が発生した場合、強制的にデータベースとの接続が解除され、データベースの更新が行えなくなるといったことである。このために、接続状態の確認や再接続の仕組みを取り入れるなどの対策を行う必要があった。

4. 障害検知と復旧処理

4.1 障害検知とサーバの切替え

前章まで、信頼性を向上させるためのサーバ間でのデータの同期方法について述べてきたが、障害発生時もしくは障害からの回復時に適切にデータが保持・復旧されなくては無意味なものになってしまう。そこで、本章では、サーバ間の障害検知方法と復旧処理について述べる。

障害発生時の処理を適切に行うためにまず必要となるのが、お互いのサーバの状態を監視し障害の発生を検出することである。これには独自のアプリケーションによる方法とSNMP（Simple Network Management Protocol）などを用いる方法が考えられる。しかし、障害の発生をトリガーとして様々な処理を自動化する必要があるため、ここでも独自の手法を用いることとした。障害検知処理は、図4で示した状態監視プロセスが行っており、自サーバの状態を監視しながら、他サーバの状態を共有することにより実現している。こうすることにより障害が発生した場合に、どちらか生存しているサーバが障害を検知でき、分析クライアントに対して障害の発生を通知することも可能になる。

障害を検知した後、サーバは単体運用へ移行するための処理を行う。サーバ切替え処理では、障害が発生したサーバに割り当てられていたIPアドレスを、正常なサーバにエイリアスIPアドレスとして割り当てる処理などを行い、端末や分析クライアントにサーバの運用状態を意識させないための準備を行う。また、切替え処理中に単体運用時の障害に備えたバックアップの作成なども行うため、他のプロセスの稼動を停止しており、この間は顧客へのサービスを提供することができない。

4.2 復旧処理

単体運用状態のサーバでは通常時と同等の機能を提供するものの、単体運用でのサーバ障害ではデータの保証が限定されるため、早急にサーバの修復(交換)を行う運用としている。サーバの交換時は、次に述べる復旧処理を行うことにより、通常の運用状態に回復する。

復旧処理は故障が発生したサーバを修理もしくは交換した後に、再び二重化構成の運用に戻すための処理である。主な目的は、単体運用時に収集・更新したデータを、交換したサーバに反映することと、停止しているレプリケーション処理など二重化システムに必要なプロセスを再開することである。復旧処理を行うためには、InterBaseの停止や起動、サーバのIPアドレスの切替えなどLinuxおよびInterBaseの操作が必要になる。しかし、「Linuxやデータベースの知識がなくても操作できる」というシステム要件を満たすため、これらの操作をすべて自動化し、ユーザは処理を選択することだけで実行できるようにしている。データの反映は、データベースの完全なバックアップを作成し、もう一方のサーバでこれを展開することにより行う。この間は、復旧処理を確実に行うために他のプロセスの稼動を停止しデータの更新が行われないようにしている。しかし、このために復旧処理を行うには本システムが提供するサービスを全て停止しなければならないという制約がある。

5. おわりに

本稿では、P-BANK SQUARE Vに搭載したサーバ二重化の構成および実装方法について、検討した内容と、データベースの持つレプリケーション機能を使用せずに独自の方法を実装した事例を紹介してきた。3章で述べたようにレプリケーション処理は、他のプロセスとは非同期に動作することにより、複数プロセスの独立性を維持しながら、データの整合性を向上させている。このことは、プログラム開発や保守の面から考えた場合、それぞれがレプリケーションを意識する必要がないため、非常に有効な構成になっている。

サーバの切替え処理や復旧処理では、独自のアプリケーションを開発して処理を自動化したことによりユーザのスキルに関係なく実行することが可能となった。しかしながら、現時点では、これらの処理を行うにあたり、サービスを一時的に停止させる必要があるため、わずかではあるが正常な営業ができない時間が存在する。このような点を、システムとしてどのように対応し、回避・軽減するかについて検討することが今後の課題として挙げられる。

本稿において、Linuxサーバを二重化し、実用に耐え得るレプリケーション処理を実現できることを示せたと思う。しかしながら、独自の手法を開発することがコスト面から見て可能であったのは、本システムが数多く出荷されるシステム製品であるという前提があったためである。Linuxを用いた二重化システムを導入・構築する際には、Linuxを採用することの強みのひとつである「オープンソースソフトウェア」を積極的に利用することなどを含め、求められる要件に対する実現性や開発コストについて慎重に検討する必要があると考える。

参考文献

(1) W.リチャード スティーヴンス(篠田 陽一 訳):“UNIXネットワークプログラミング〈Vol.2〉IPC:プロセス間通信”, ピアソンエデュケーション,(2000)

(2) W.リチャード スティーヴンス(大木 敦雄 訳):“詳解UNIXプログラミング”, ピアソンエデュケーション,(2000)

中島 龍太朗
Ryutaro Nakashima

・関西地区本部
アドバンスト・テクノロジー・システム部
・Linux 上で稼動するサーバアプリケーションの開発に従事

大和 浩之
Hiroyuki Yamato

・関西地区本部
アドバンスト・テクノロジー・システム部
・アプリケーション開発のプロジェクトマネジメントに従事