業務停止はさせない！
トラブル時は自動切替えで PostgreSQL の運用を継続
－富士通の技術者に聞く！PostgreSQL の技術－

企業の生産活動を支える ICT システムの停止は、いまや生産活動そのものの停止につながるといっても過言ではない。トラブルが起きた際に業務をいかに継続できるか？はデータベースにおける信頼性の重要な要素である。それは「PostgreSQL」をエンタープライズ利用する上で大事なポイントの 1 つでもある。富士通では、これに対する解として Symfoware Server に搭載されている「富士通版 PostgreSQL」に、確実な業務継続を目指す「データベース多重化機能」を実装した。ここでは技術者がどのように機能を検討し開発したのか？縮退を自動で行ない復旧はコマンドひとつで OK という手軽さは一体どのように実現したのか？その真相に迫る。

谷口 康規 Yasunori Taniguchi

富士通株式会社　ミドルウェア事業本部 データマネジメント・ミドルウェア事業部
専門分野：データベース
20 年あまり旧 Symfoware や PostgreSQL ベースの Symfoware の開発に携わり、SQL 言語処理からデータロードコマンド、リカバリー、GUI に至るまで様々な視点で機能開発を経験してきました。一方で、オープンソースへのシフトや IoT への時代の変化を見てきました。このようなデータベースを取り巻く劇的な環境の変化がある中で、従来の開発経験を照らし合わせると、データベースは何と地味で変化のない制御系ソフトウェアであろうか！と思っていました。しかしながら、昨今では PostgreSQL は多方面の分野で利用され、練られた内容が積極的に機能に取り込まれていっている大いに期待のできるデータベースであると思います。そのような PostgreSQL をベースとした Symfoware にもさらなる発展が期待できます。ご期待ください！

鎌内 彬貴 Akitaka Kamauchi

専門分野：データベース

システム規模と価格帯をマッチさせるために

「WAL が損失するという課題の解決」の回で「信頼性」に関わるポイントの 1 つである「データ保全」の機能について伺いました。「業務継続」は同じく信頼性に関わるポイントですが、それを実現するデータベース多重化機能とはどのような機能なのでしょうか？

- 【関連特集】WAL が損失するという課題の解決

鎌内

データベースは ICT システムを支える非常に重要な役目を担っています。このため、データベースの停止は、システム管理者にとって避けなくてはならない重要なテーマです。データベースが停止すると、それを利用している業務システム全体が停止してしまいます。しかし「形ある物は必ず壊れる（ハードウェアの故障）」し、「人間はミスをする生き物（ソフトウェアトラブル、ヒューマンエラー）」なので、トラブルを完全に避けることはできません。このような状況から業務システムの停止を避けるために、データベースの複製をリアルタイムに作成しておき、何らかのシステム異常を検知すると自動的に複製されたデータベースに切り替えて業務システムの運用を継続できるようにするのが、当社の商用データベース「Symfoware Server」に搭載された PostgreSQL で提供しているデータベース多重化機能です。

オリジナルの PostgreSQL には共有ディスク方式のクラスタリング機能がありますよね？それにもかかわらずデータベース多重化機能を開発したのはなぜでしょうか？

谷口

確かに共有ディスク方式のクラスタリング機能でもデータベースの可用性を高めることはできます。しかし、共有ディスク装置はそれなりに高価です。また、今でこそ大規模システムでも PostgreSQL が活用されるようになりましたが、PostgreSQL を活用するシステムの多くは中小規模です。このような中小規模のシステムでは共用ディスク装置の価格帯はマッチしないと感じていました。このため、システム規模と価格帯がマッチする可用性の開発が必要だと考えたのです。

それがデータベースを丸ごと複製するという機能だったのですね。

鎌内

そのとおりです。データベースを複製するのであれば、共有ディスク装置のような特別なハードウェアは不要です。例えば社内に 2 台のサーバーが存在していたら、「これを使って高可用システムが構築できる」という感じで手軽に構築できるようになればよいと考えました。それと「運用性／使い勝手としてのオープン」を実現するうえで、クラスタリング機能における導入や運用の複雑さにも着目しました。

クラスタリング機能における導入や運用の複雑さですか？

谷口

はい。一般的にクラスタリング機能を使うには、データベースソフトウェアに加えてクラスタリングソフトウェアが必要となり、両方をインストール・セットアップしなければなりません。さらに共用ディスク方式の場合は、プライマリーサーバーとセカンダリーサーバーの双方から共有ディスクをアクセスできるように設定する必要もあります。また、システム異常が発生した際の復旧には高度な技術が必要です。データベースソフトウェアとクラスタリングソフトウェアはとても密接な関係にあるため、それぞれの復旧作業を手順に従って正しく実施しなければなりません。この手順を間違えてしまうとシステム停止につながり、せっかくの可用性を活かせないということが起こります。この点、私たちが開発したデータベース多重化機能であればクラスタリングソフトウェアが不要になり復旧手順もシンプルになります。

ストリーミングレプリケーションに機能をプラス

手順がシンプルであれば、システムトラブルのリスクは軽減できますね。ところで、オリジナルの PostgreSQL ではバージョン 9.0 からストリーミングレプリケーションが実装されていますが、違いは何でしょうか？

谷口

違いと言いますか、そもそもデータベース多重化機能はオリジナルの PostgreSQL に実装されているストリーミングレプリケーション（注 1）の機能を利用して実現しています。しかしストリーミングレプリケーションは、データベースを冗長化する機能にのみ特化しています。そのため、システムの異常を検知して自動的に切り替える部分はアプリケーションで実現する必要があります。この異常検知や自動切り替えにはそれなりのノウハウが必要なため、容易に導入できるとは言い難いのが現実です。そして、ストリーミングレプリケーションにはまさにこの部分が不足しており、私たちが考えるエンタープライズ利用に対する機能が足りていないと感じていました。データベース多重化機能は、ストリーミングレプリケーションの機能に、当社のノウハウを結集した異常検知機能および自動切替え機能をプラスして Symfoware Server の PostgreSQL に組み込んだものであり、業務継続性と運用性を格段に向上させています。

注 1　PostgreSQL 9.0 から実装されたデータベースの複製機能。プライマリーデータベースの更新内容をセカンダリーデータベースへ WAL レコード単位に転送してデータベースを複製（レプリケーション）することができる。ファイル単位などの大きな塊ごとに転送するのではなく、データを次々と流れる（ストリーム）ように転送することからストリーミングと呼ぶ。

ストリーミングレプリケーションの機能強化版ということですね。もう少し具体的に教えてください。

鎌内

実際にサーバーを切り替えて業務を継続するためには、どのような障害が起きたのかを検知しなければなりません。障害には OS またはサーバーがダウンしたのか？無応答なのか？あるいは、データベースのダウンや無応答、さらにはハードディスクの物理的な故障でデータが読み取れないなど、あらゆる事象が想定されます。この「どこの箇所で、どのような異常が起きているの

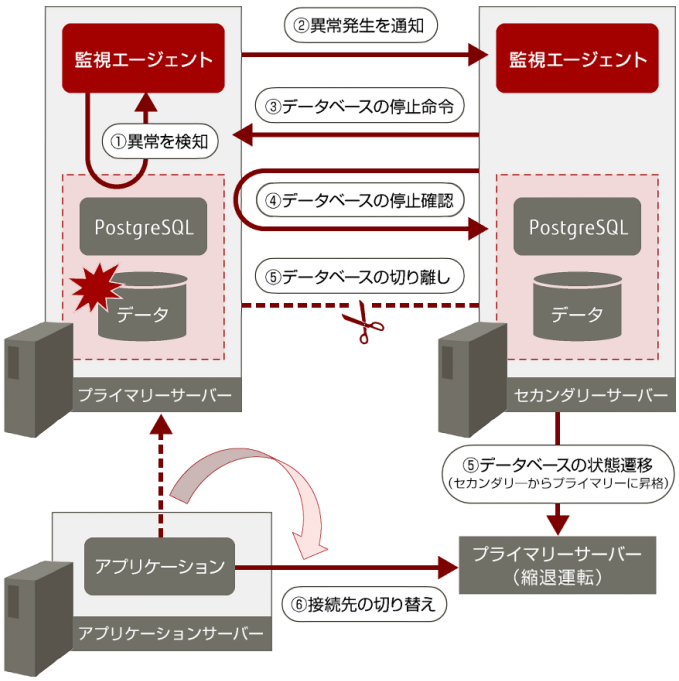
か？」を検知するエージェントプログラムをプライマリーサーバーとセカンダリーサーバーに配置し、データベースを監視・制御しています。これら監視制御技術は、Symfoware Server で提供しているデータベースミラーリングによる完全二重化システムの考え方に基づいて、異常検知技術や自動切替え技術を応用したものです。

- Symfoware Server 製品情報
  https://www.fujitsu.com/jp/products/software/middleware/database/symfoware/products/symfowareserver/

ノウハウが必要な監視と制御の部分に実績ある技術が使われているのですね。実際どのように切り替えが行なわれるのですか？

谷口

プライマリーサーバーに何らかの異常が発生するとエージェントプログラムがそれを検出し、セカンダリーサーバーのエージェントプログラムに異常が発生したことを通知します。通知を受けたセカンダリーサーバーのエージェントプログラムは、プライマリーサーバーにデータベースの停止を命令します。プライマリーサーバーのデータベース停止が確認できたら、セカンダリーサーバーのデータベースが新しいプライマリーサーバーのデータベースになるよう状態遷移（セカンダリーからプライマリーに昇格）させます。このときに、異常が発生した元プライマリーサーバーを切り離してシステムを縮退運転させるといった流れになります。



図：サーバー切り替えの遷移

異常が発生したサーバーを切り離すのですか？

鎌内

速やかに業務継続するには、異常が発生しているサーバーの影響を他に広げないようにすることが重要です。そのためには一刻も早く業務システムから切り離す必要があるのです。データベース多重化機能では、障害の起きたサーバーを切り離して複製されているサーバーに丸ごと切り替えるため最新のデータをそのまま利用できます。

しかし縮退運転となるとアプリケーションは接続先の切り替えを意識する必要がありませんか？

谷口

アプリケーションからは物理的なサーバーを意識せずに接続ができる「透過的接続」を実現しているため、切り替えを意識する必要はありません。もちろん、アプリケーションの利用者も切り替えを意識する必要はありませんし、切替え時間は秒オーダーで済みます。アプリケーションからデータベースを見た場合、異常が発生するとトランザクションはエラーで返却されます。こ

のトランザクションはリトライされるわけですが、このときデータベースクライアントがプライマリーサーバーやセカンダリーサーバーを自動的に識別してデータベースに再接続しますのでアプリケーションは何も意識しなくてよいのです。

ここでも「透過的＝アプリケーション開発者が何も意識をする必要がない」を実現しているのですね。

復旧手順をわかりやすく

先ほど、トラブル発生時にはサーバーを切り離して縮退運転を行なうと伺いましたが、縮退運転からは早急に復旧することが大切だと思います。この点について、何か取り組まれているのですか？

谷口

復旧手順の簡易化を実現しています。サーバーの切り替えは自動的に行なわれるのですが、復旧は人が作業しなければなりません。この大事な作業において、複雑な手順というのは二次的なミスにつながりかねません。このため、復旧手順をできるだけ少なく、分かりやすい操作になるように設計しました。

手順自体が少なければミスもその分減らせますね。具体的にはどういう手順なのでしょうか。

鎌内

最初にトラブルの基となった原因を解決する作業を行ないます。ハードウェアが故障したのであれば該当するハードウェアを新しいものに取り換えたり、プログラムがダウンしたのであれば修正やパッチの適用、パラメーター修正などの処置を行なってから該当するプログラムを再起動したりといった作業です。その作業が済んだら、切り離した元プライマリーサーバーを復旧させるのですが、その操作は"コマンドひとつ"で終わります。

コマンドひとつですか？

谷口

はい、コマンドひとつです。ストリーミングレプリケーションにおける復旧作業では、プライマリーサーバーとセカンダリーサーバーに存在している環境設定ファイル「postgresql.conf」の内容を修正したり、新しく組み込み直すセカンダリーサーバーにリカバリー設定ファイル「recovery.conf」を作成したり、異常発生前のディレクトリ構成を復元するといった、それぞれのサーバー間を行き来するいくつかの作業があります。データベース多重化機能では、切り替え後のプライマリーサーバー（元セカンダリーサーバー）に存在している各種設定やデータと同期させるといったこれらの作業をコマンドひとつに集約しました。

開発の苦労は？

複数の作業をコマンドひとつにまとめるということは容易ではないように思いますが、開発は苦労されたのではないでしょうか？

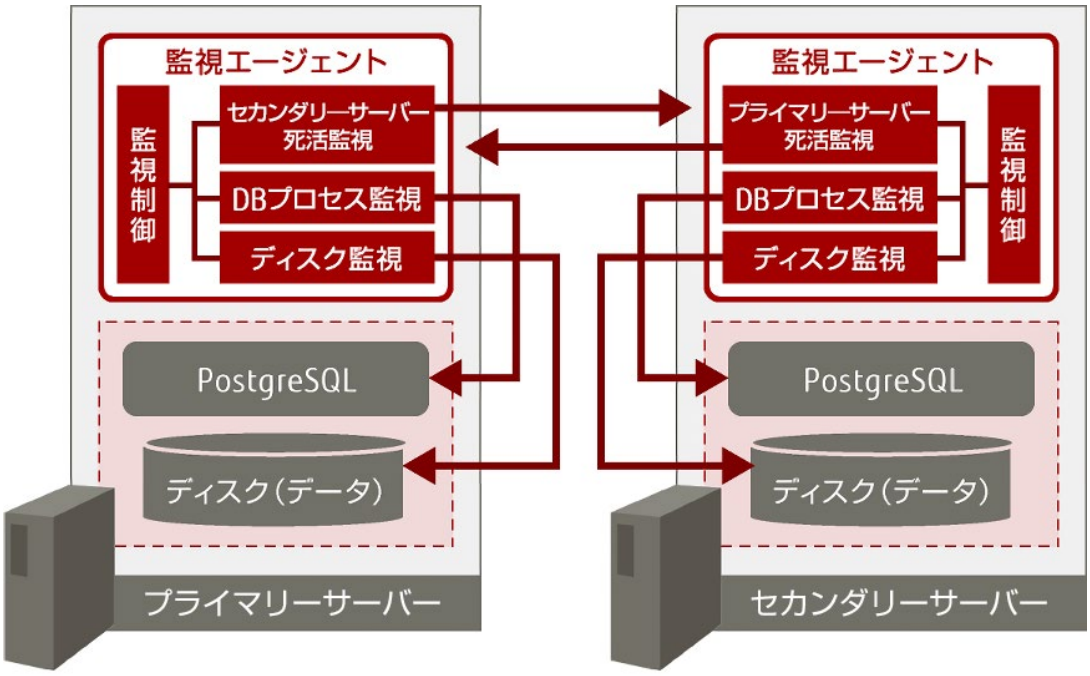
鎌内

復旧コマンドの仕様については、かなり議論を重ねました。「コマンドひとつで」というコンセプトはすぐに決定しましたが、いざ仕様を決める段階になるとオプションをどこまで用意するべきなのか悩みました。 きめ細かに設定できた方がよいと考えると指定できるオプションは増えていきます。しかしそれは設定項目が増えることにつながり、ひいては複雑になってしまう原因となるので当初のコンセプトと"ずれ"が生じます。逆にオプションを極力削ってしまうと簡単にはなりますが、そもそも本来の機能としてどうなのか？というところにつながってしまうので落としどころを見つけるまでに議論を重ねました。

使いやすいと簡単に見える印象がありますが、その裏にはいろいろな苦労があるのですね。

谷口

そうですね、特に異常検知機能と自動切替え機能の開発作業は苦労しました。一口に「異常」といってもさまざまなパターンがあります。異常を検知するため各サーバーに対して、データベースプロセス監視、ディスク監視、ペアとなるサーバーの死活監視（OS の監視）を担う 3 つのスレッドを用意することで、自分自身と相手を常に監視するようになっています。そしてこの 3 つの監視スレッドを制御しながら、ペアとなる相手サーバーとの整合性を保証する制御機構も用意することで、確実に異常を検知できるように設計しています。加えて、エラー事象ごとに「切り替え対象とするのか？」「切り替えは行なわずエラーを通知するだけにするのか？」といった分類にも非常に時間をかけました。



図：異常を検知するさまざまな仕組み

先ほど、異常検知と切り替えのノウハウは Symfoware Server から応用したと伺いましたが、その開発時にエラー事象の分類を実施したのではないのですか？

鎌内

もちろん、実施しました。しかし、データベース多重化機能は先に開発された Symfoware Server のデータベースミラーリング機能とコンセプトや実現する機能は同じなのですが、ベースが PostgreSQL のため全て流用できるとはいきませんでした。正確にいうとコンセプトを基にしてゼロから作りこむイメージです。なぜなら、PostgreSQL の挙動は事象ごとに Symfoware Server と異なります。このため、パターンをマトリックス化して 1 つ 1 つプログラムを追うだけではなく、実際に事象を再現して確認しながら開発しました。これはとても大変でした。

そういった地道な作業が"簡単に運用できる"というところにつながっているのですね。今後強化したい点はありますか？

谷口

強化したい点は 2 つあります。1 つは「信頼性のバリエーション強化」です。現在、データベース多重化機能で実現しているのは Symfoware Server に搭載されているデータベースミラーリング機能と同様の 1 対 1 完全二重化システムです。もちろん、この 1 対 1 でも信頼性は十分に確保できると考えています。例えば、データベース多重化機能と遠隔地レプリケーションを組み合せて運用することで、災害対策も兼ねた多重故障に対応できるシステムを構築することもできます。ただ、信頼性の要件はシステム用途に応じていろいろあります。現にオリジナルの PostgreSQL に搭載されているストリーミングレプリケーションは 1 対 N に対応しています。これを活用して今後は 1 対 N のデータベース多重化機能、つまり 1 台のプライマリーサーバーのデータベースを複数

台用意されたセカンダリーサーバーのデータベースに同期反映できる機能を実現したいと考えています。これにより、利用者の多重故障に対する信頼性の選択肢を増やしていきたいと考えています。

信頼性の選択肢が広がれば、PostgreSQL の活用範囲もさらに広がりますね。もう 1 つはなんでしょう。

谷口

「導入のしやすさ」を向上することです。CUI だけでなく、GUI による視覚を活用した直感的な操作で、データベース多重化機能のセットアップ作業や復旧作業を実施できるようにしたいと考えています。データベース多重化機能はストリーミングレプリケーションをベースにしているため、ストリーミングレプリケーションのセットアップに関する知識が必要です。残念ながら、現状ストリーミングレプリケーションのセットアップは少々複雑なのです。このセットアップ作業が GUI で行なえれば、これまでストリーミングレプリケーションをセットアップしたことのない技術者でも簡単にセットアップできるようになります。同様に、サーバーの状態確認や復旧作業も GUI で直感的に操作できれば、データベース専任の作業者でなくても運用できるようになります。このように「運用性／使い勝手としてのオープン」を具現化した、誰でも容易に高信頼なシステムを利用できることを目指した運用性向上に取り組んでいきたいと考えています。

エンタープライズ用途であっても「運用性／使い勝手としてのオープン」を目指すことで人に優しいソフトウェアを実現していくのですね。

2015 年 12 月 4 日