PostgreSQL の高可用性を追求！

Connection Manager でデータベースへの接続を瞬時に切り替える

－富士通の技術者に聞く！PostgreSQL の技術－

データベースシステムの可用性向上のためにはシステムを冗長化し、サーバーに障害が発生した場合においては業務に支障がないよう迅速にスタンバイサーバーに切り替えることは非常に重要です。そのためには、サーバー側だけでなくアプリケーション側の考慮も必要であり、どんなに早くスタンバイサーバーに切り替わってもアプリケーションからの接続先の切り替えに時間がかかっては意味がありません。PostgreSQL のクライアントドライバーにも可用性を考慮した、アプリケーションとサーバー間との接続を維持するための機能が実装されていますが、FUJITSU Software Enterprise Postgres 12（以降、Enterprise Postgres と略す）では、その接続性をさらに強化した機能「Connection Manager（コネクション マネージャー）」を提供しています。
本特集では、Connection Manager 機能の開発担当である「松村 量」が、機能をわかりやすく解説するとともに、機能開発への思いや将来に向けた展望について語ります。

松村 量 Ryo Matsumura
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
入社以来、データベースに携わってきた。富士通が独自開発した Symfoware Server のトランザクション管理や復旧機能の開発を経て、現在は PostgreSQL をベースとした「FUJITSU Software Enterprise Postgres」の様々な機能を開発している。

Connection Manager とは何か

はじめに、Connection Manager とはどういうものなのか、簡単な紹介をお願いします。

松村

データベースシステムにおいて、システム規模が大きくなればなるほど業務継続への要求は高くなります。この要求に対して PostgreSQL では、レプリケーションというシステム構成をとることで、万が一に備えた対策を講じるのが定石です。さらに、別サーバー上で稼働しているアプリケーションとデータベースサーバーとの接続を維持することも考慮しなくてはいけません。このアプリケーションとデータベースサーバーとの接続を制御するのが Connection Manager です。アプリケーションがどのサーバーにつながっているのか、また障害が発生した場合にどのサーバーに切り替えるのかといった接続に関する制御を高速に行います。

Connection Manager とは、「アプリケーションとデータベースとの間の接続を監視 / 制御する機能」なのですね。

松村

はい、Connection Manager は、アプリケーションとデータベースサーバーをつなぎ、可用性を向上させることで安定した業務継続を実現する機能です。大きく 2 つの機能から構成されています。
1 つは、アプリケーションの透過的接続を行う機能です。例えば、レプリケーション構成の場合、アプリケーション側からデータベースサーバーに接続するときに、指定された属性をもつサーバーに速やかに接続します。属性とはレプリケーション構成のプライマリーやスタンバイのことです。
もう 1 つは、アプリケーションとデータベースサーバーの生死監視機能です。アプリケーションが動作するサーバーとデータベースサーバー間を相互に監視することで、各サーバーで異常が発生したりネットワークの健全性が損なわれたときに、アプリケーション側とデータベースサーバー側に異常を通知します。

アプリケーションから速やかに接続するための仕組み

では早速、それらの機能についてお聞きしていきたいと思います。まず、アプリケーションの透過的接続機能ですが、既に多くのクライアントドライバーで、接続時にサーバー属性を指定するといった同様の機能が実装されているように思います。これとは、どこが違うのでしょうか？

松村

例えば、PostgreSQL のクライアントドライバー「libpq」で使用する target_session_attrs パラメーターが類似機能に相当します。クライアントドライバーで実装されているこのような機能を使った接続で、レプリケーション構成へ接続しようとすると、データベースサーバーやネットワークの状況、タイムアウトの設定時間によっては、速いときと遅いときとで何倍もの差が出てしまいます。データベースシステムにおいて、接続時に要する時間の不安定さは致命的です。そこで、Connection Manager では、「一定時間内に速やかに接続ができること」、また接続に失敗した場合も「一定時間内にエラーが通知されること」を目指しました。

接続を安定させるということですね。それぞれの仕組みの違いを教えてください。

松村

まず、クライアントドライバーによる実装では、ユーザーが指定した接続情報（IP アドレスとポート番号）をもとに、そこに記載された記述順に接続を試行します。順番に接続してサーバーの属性を確認する必要があるということです。仮に、サーバーA とサーバーB の 2 台で構成されたレプリケーションシステムがあり、「サーバーA がプライマリー」、「サーバーB がスタンバイ」の属性を持つと仮定します。ユーザーが指定した接続順は、「サーバーA、サーバーB」の順とします。ここで、クライアントドライバーがプライマリーの属性を持つサーバーに接続しようとした場合、通常の状態では、サーバーA に接続されるため何も問題はありません。しかし、何らかのトラブルでプライマリーがサーバーB に切り替わってしまった場合でも、サーバーA から順に接続を試みるため、接続に時間がかかってしまいます。

不要な接続処理が発生するわけですね。

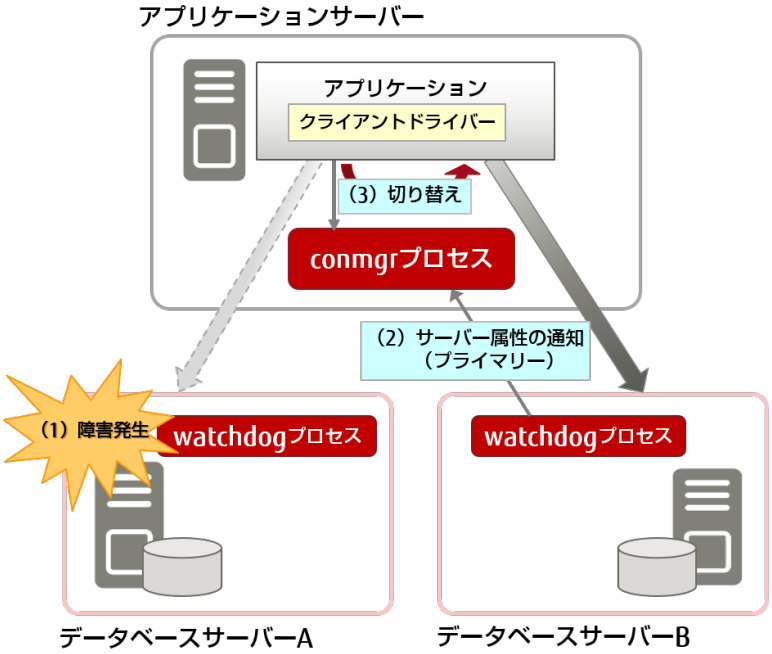
松村

そのとおりです。この問題を解決するのが、Connection Manager なのです。Connection Manager は、アプリケーションサーバーに専用プロセス（conmgr）を常駐させています。このプロセスは、接続先のすべてのデータベースサーバーの属性や状態を常に監視するプロセスです。ユーザーは、アプリケーションからデータベースサーバーへの接続時に、この conmgr プロセスに接続するための情報（ポート番号など）を指定します。アプリケーションは、指定された conmgr プロセスに接続することで、データベースサーバーの IP アドレスとポート番号を知ることができます。その後は、その IP アドレスとポート番号を使って、アプリケーションが直接データベースサーバーに接続するというわけです。この仕組みにより、接続に要する時間も最小限にとどめることができます。

conmgr プロセスは、データベースの属性変更があった場合、例えば、現在、どのサーバーがプライマリーなのかということをどのように認識するのでしょうか？

松村

先ほど、アプリケーション側に conmgr プロセスが存在すると説明しましたが、データベースサーバー側にも watchdog プロセスという専用プロセスを常駐させています。この watchdog プロセスはサーバーの属性を常にチェックしており、その属性を conmgr プロセスに通知します。この仕組みにより、conmgr プロセスはサーバーの属性が変更されたことをタイムリーに知ることができます。



他にも似たような機能を提供する PostgreSQL の OSS ツールなどがあると思いますが、これらとはどこが異なるのでしょうか。

松村

Pgpool-II や pgbouncer ですね。これらは、アプリケーションとデータベースサーバーの間で稼働するプロキシソフトウェアです。アプリケーションとデータベースサーバーとの通信をこれらのソフトウェアが中継しています。これによって、コネクションプーリングが可能になったり、ロードバランス（SQL の内容を解析することでアクセスするデータベースサーバーを振り分ける）などの様々な機能を生み出すことができます。その一方で通信の中継によってレイテンシーが悪化することを免れません。
これに対して、Connection Manager では、コネクションプーリングやロードバランスなどの高度な機能を提供しない代わりに、アプリケーションとデータベースサーバーを直接、接続します。SQL のレスポンスの素早さを重視する場合には、Connection Manager を使っていただき、コネクションプーリングなどの機能を利用したい場合には Pgpool-II や pgbouncer を使っていただく、というように棲み分ける関係であると理解していただけると良いと思います。

Enterprise Postgres のクライアントドライバーは、ユーザーによって指定されたサーバーに接続するだけではないのですね。これは、PostgreSQL のクライアントドライバーの動作とは違うように思います。

松村

はい。この動作のために、オリジナルのクライアントドライバーを改造しています。Pgpool-II などのオープンソースソフトウェアでは、このようにクライアントドライバーの改造を必要とするようなアーキテクチャーを採用することはできません。しかし、Enterprise Postgres は、PostgreSQL のすべてのクライアントドライバーも同梱するので、このような高速レスポンスを実現できるアーキテクチャーを採用できるのです。

サーバー間のハートビートで異常を検知

では、もう 1 つの機能である生死監視機能について教えてください。

松村

この機能は、アプリケーションが動作するサーバーとデータベースサーバーの相互監視を行うことで、長時間、無応答になってしまうようなネットワーク不通やサーバーダウンを素早く検知して、データベースサーバー側でのリソース回収とアプリケーションに対するエラー返却を行うものです。

この機能には、具体的にどのようなメリットがあるのですか？

松村

アプリケーションとデータベースサーバーとは、TCP で通信していますが、プロトコル上は、通信やサーバーの異常を検知する仕組みが存在しません。そのため、相手側が通信の閉鎖手続きを取らない限りは、相手が通信を続ける可能性があると判断しなければなりません。
例えば、相手側がオペレーションシステムごとダウンしていたり、あるいは、ネットワークが不通となっている場合でも、閉鎖手続きが取られていないので、その相手との通信のための環境を維持しておかなければならないということです。これを、アプリケーションとデータベースサーバーとの通信に当てはめると、アプリケーション側ではデータベースサーバー側から SQL の応答が返る可能性を信じて待ち続けることに相当します。データベースサーバー側では、アプリケーションからのクエリを待つために必要なメモリを確保し続けたり、トランザクションをオープンしたままにすることに相当します。
これに対して、Connection Manager は、通信やサーバーに何らかの異常が生じた場合、所定の時間内にそのことを検知し、クライアントドライバーを通してアプリケーションに異常を通知します、また、データベースサーバー側では SQL 接続を強制回収するため通信の環境を維持し続けておく必要がなく、アプリケーション側からは直ちに接続のリトライが可能となるわけです。

異常が発生した場合に接続を維持し続けることがなくなるわけですね。しかし、TCP 通信での異常検知には、keepalive 機能が有効であるように思います。

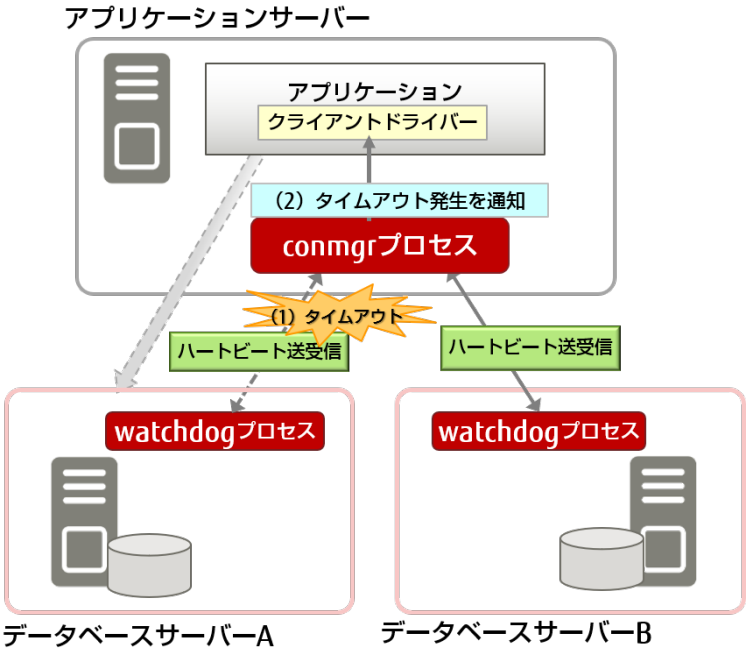
松村

keepalive 機能を使用する場合、本来ならば、keepalive 機能を有効にするだけでなく、オペレーティングシステム全体に影響が及ぶ TCP パケットの再送回数や時間の上限値の設定も必要なのですが、適切な値を設定することが難しく、設定自体が見過ごされることが多いです。keepalive 機能を有効にするだけでは、例えば、アプリケーションによる SQL 文の送信中にネットワーク切断などを検知できないのです。これらを設定せずに異常を検知するには、アプリケーション層で異常検知のプログラムが不可欠になりますが、Connection Manager がこれを担うということです。

keepalive 機能で異常が検知できない問題を Connection Manager ではどのように解決したのですか？

松村

異常を検知するための一般的な方法としてハートビートと呼ばれる手法があります。心臓の鼓動（ハートビート）のように、サーバー間で相互に、一定間隔でデータを送信しあい、このデータを一定期間内に受信できなければ相手のサーバーがダウンしたか、ネットワークが不通になったとみなす手法です。しかし、この手法をクライアントドライバーに適用することは困難が伴います。なぜならば、ユーザーのアプリケーションロジックとは非同期にハートビートで通信しあう必要があるからです。そこで、Connection Manager では、さきほど登場した conmgr プロセスと watchdog プロセスとの間でハートビート送受信を行うようにしました。これにより、アプリケーションロジックと独立させることができるわけです。そして、Enterprise Postgres のクライアントドライバーと conmgr プロセスは常に接続を保持しているので、conmgr プロセスがハートビートでタイムアウトを検知するとすぐにクライアントドライバーに異常が通知されます。



watchdog プロセスと conmgr プロセス間のハートビートと、アプリケーションと conmgr プロセス間で保持された接続が Connection Manager の特長的な仕組みであると感じました。

松村

はい、先ほど、アプリケーションとデータベースサーバーが直接、接続することで高レスポンスを実現していることをお話ししました。このとき、アプリケーションとデータベースサーバーとの接続に conmgr プロセスが経由されていないので、conmgr プロセスで異常が検知された場合、それをアプリケーションにどのように伝えるのかがポイントです。この異常の通知にアプリケーションと conmgr プロセス間の接続が使用されているわけです。

データベースの監視と接続先の決定を Connection Manager に任せることで、アプリケーション開発者の負担が大きく軽減できますね。最後に、今後の取り組みなどを教えてください。

松村

Enterprise Postgres 12 では、Connection Manager に対応したクライアントドライバーは、libpq ライブラリーと、これを使用する C 言語の埋め込み SQL（ECPG）、COBOL 言語の埋め込み SQL（ECOBPG）、psql コマンドに限られています。今後は、JDBC ドライバー、ODBC ドライバー、Npgsql などでも利用できるようにする計画です。また、現在は、適応するプラットフォームも Linux のみのため、Windows など他のプラットフォーム展開にも取り組みます。

クライアントドライバーやプラットフォームに限定されず、今後は、さまざまなアプリケーションにおいても可用性向上につながりますね。高信頼を追求する Enterprise Postgres において、その一翼を担う機能ということで、これからも注目です。ありがとうございました。

2020 年 8 月 28 日