一日前のデータはもう古い!?
大量データもリアルタイムに意思決定

－富士通の技術者に聞く！PostgreSQL の技術－

「インメモリ」「列指向（カラムナ）」などのデータベース技術が実用化され、OLAP の世界に数々のブレイクスルーをもたらしている。富士通では、「世界一高速な PostgreSQL をいち早く世に提供すること」をコンセプトに、独自の「インメモリカラムナ」技術を開発し、トランザクション処理の一貫性を保ちながら、分析処理の高速性を透過的なインターフェイスで実現した。その真相に迫る。

河場 基行 Motoyuki Kawaba

専門分野：性能分析、性能解析、データベース

より多くのデータをより早く処理するために生まれたインメモリカラムナ機能

「インメモリカラムナ」というキーワードは、近年データベースでよく耳にします。PostgreSQL にインメモリカラムナを開発された目的についてお聞かせください。

河場

ビッグデータ活用が本格化し、IoT データや、オープンデータ、SNS など、これまでの業務システムで使用していなかった新しいデータの活用が注目されていますが、従来から情報分析に使われていた基幹システムの業務データは、現在も情報分析に無くてはならない存在です。基幹システムの業務データは、よりリアルタイムに参照・活用できるスピードが求められています。たとえば「午前の売上データの速報値を見て午後の対策を決定する」など、意思決定のスピードが急速に変化してきています。

それは、これまで夜間のバッチ処理や、データウェアハウスで行ってきた大量データ集計のスピードが変わってきたということですか。

河場

はい。従来はオンライントランザクション処理（OLTP）の性能に影響を与えないよう、時間帯を変え夜間に集計したり、データウェアハウスなど別システムで集計したりしていました。そのため、参照できる最新データは前日のデータということが殆どでした。しかし、先ほどお話したとおり、もはやそれではビジネスニーズを満たせないケースが出てきているのです。このニーズに対応するには、OLTP と大量データ検索・集計処理、つまりオンライン分析処理（OLAP）が同時にできる、というデータベース自体の進化が重要になってきます。既に数年前から、商用データベース各社が OLTP と OLAP の共存に向けた機能を提供してきています。そして、この共存の流れは今後の PostgreSQL にも重要な要素になると考え取り組んだのが、今回のインメモリカラムナです。

行形式と列形式のデータを共存させるインメモリカラムナ機能の仕組み

PostgreSQL にインメモリカラムナ機能を新たに加えたのは、こうした背景があったからなのですね。富士通のインメモリカラムナ機能の具体的な仕組みについて、簡単に教えていただけますか？

河場

この機能の中核技術の 1 つが「カラムナ」、つまり「列形式」でデータを格納する技術です。一般的な RDBMS は「行形式」でデータを格納しますが、これはデータの更新・削除には強いものの、大量データの検索処理の効率は決してよくありません。逆に、データ更新・削除はあまり得意ではないものの、大量データの検索処理にめっぽう強いのが列形式です。そこで、OLAP 用途を前提としたデータベースの製品の多くは、列形式でデータを格納する方式をとっています。私たちが開発したインメモリカラ

 © 2016 Fujitsu Limited

ムナ機能でも、従来の行形式のデータに加えて列形式のデータも PostgreSQL で扱えるようにしたのです。具体的には、行形式のテーブルに対応するインデックスとして、列形式のテーブルを持つようにしています。

列形式のインデックスですか？

河場

はい。行形式と列形式のデータを共存させるといっても、実際の検索・集計処理で、すべてのデータが必要というわけではありません。列形式のインデックスは、行形式のインデックスと異なり、検索する可能性のある列を指定しておけば、列の組み合わせによらず高速に検索することができます。ですから、検索する可能性のある列データをインデックスとして保持できるようにしました。

「インメモリ」とついているので、列形式のインデックスはメモリー上にあるのでしょうか。

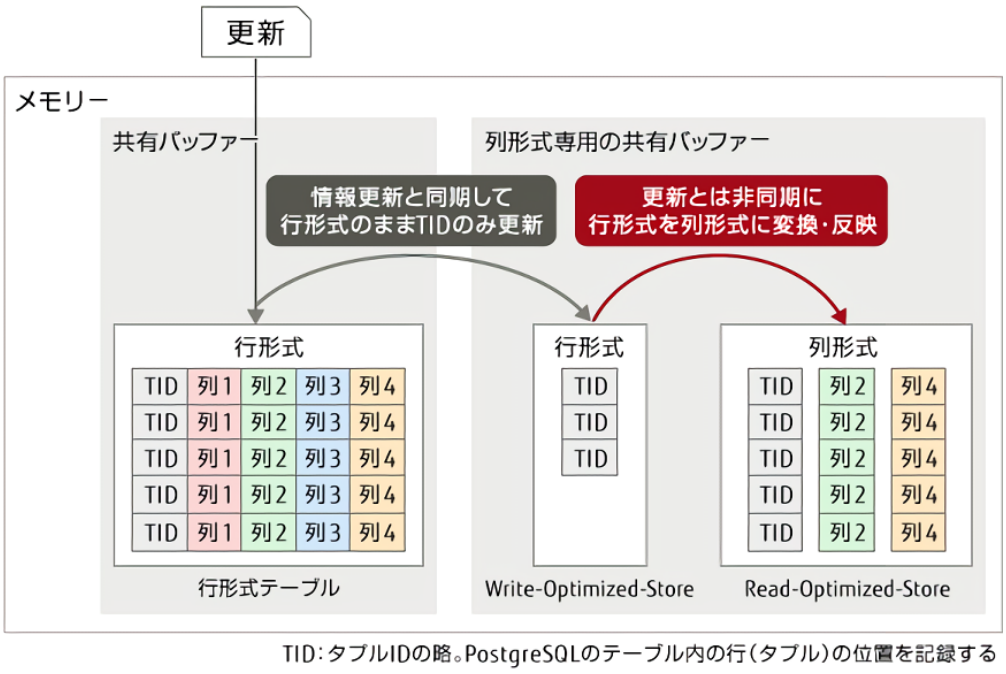
河場

はい。PostgreSQL の行形式のデータで利用する共有バッファーとは別に、列形式専用の共有バッファーを用意しました。共有バッファーを独立させることで、OLTP に影響を与えることなく大量データ集計ができるようにしています。

メモリー上の行形式と列形式のデータの内容は、同期しているのでしょうか。

河場

厳密にいうと、同期はしていませんが、同期しているのと同じ状態を保持しています。例えば、PostgreSQL に対して、データレコードを更新する命令が飛んで来たとしましょう。PostgreSQL はまず従来通り、メモリー上の行形式のテーブルに対して更新処理を行います。通常ならここで終わりなのですが、インメモリカラムナ機能を使った場合はこれと同時に、やはりメモリー上に保持した列形式のデータに対しても同様の更新処理を行うよう、命令を発行するのです。この時、実際には専用の共有バッファー上に設けた Write-Optimized-Store（以降、WOS）という追加・削除情報を記録しておく領域に、そのトランザクション情報を一時的に溜めておくのです。そしてシステムが適切なタイミングを見計らって、その内容を同じく専用共有バッファー上にある Read-Optimized-Store（以降、ROS）という領域に列形式のデータとして反映します。



図：メモリー上のデータ同期の仕組み

 © 2016 Fujitsu Limited

列形式のデータの反映は非同期なのですね。実際には非同期なのに、どのようにして同期状態を保持するのですか。

河場

データを参照する際には、ROS の列形式のデータと WOS にある差分情報を突き合わせて処理することで、結果は行形式のデータを参照した場合と同じ内容になります。

結果が同じであれば、リアルタイムに同期させなくてもよいということですね。なぜ直接同期せずに WOS に一時的にデータを溜めるのですか。

河場

行形式のデータを列形式で格納するには、データの形式を変換する必要があります。行形式のデータ更新のたびにデータを変換すると、その分オーバーヘッドが大きくなり、レスポンスに影響します。ですから、一旦行形式のまま WOS に格納することで、レスポンスへの影響を抑えているのです。また、WOS に一時的に溜めるデータも、PostgreSQL のテーブル内の行（タプル）の位置を記録するタプル ID（TID）だけを記録し、データそのものは記録しないことで、トランザクションのオーバーヘッドを最小限にしています。

処理の分離と最小限の記録で、OLTP の性能を維持されたのですね。ところで、アプリケーションからは明示的に行形式と列形式のデータの使い分けが必要なのでしょうか。

河場

アプリケーション側は、行形式データと列形式データのどちらにアクセスするかは、一切気にする必要はありません。従来通りに SQL 文を発行すれば、後はデータベースが「どちらを使った方が効率的に処理できるか」自動的に判別し、適した方のデータに自動的にアクセスを振り分けてくれます。

「二重持ち」するという選択をした、富士通ならではの仕組みやこだわり

メモリー上で行形式データと列形式データを、いわば「二重持ち」しているのですね。では、その内容をディスクに保存する際はどうしているのでしょうか？

河場

行形式データも列形式データも、両方ともディスクに保存しています。実はこの部分の実装方式は、検討を重ねました。二重持ちするとリソースは通常より多く用意しないといけません。しかし、ディスクに列形式のデータを保持していないと再起動時のメモリー再構築の時間がかかります。両者を天秤にかけ、最終的に「二重持ちする」という選択をしました。

なぜ「二重持ち」を選択したのですか？

河場

インメモリデータベース製品の中には、ディスク上には行形式でしか保存しないものもあります。しかし検討を進めるうち、クラッシュからの復旧時に列形式データがディスクに保存されていないと、メモリー上の列形式データの再構築にとても時間がかかることが判明しました。これだけ時間がかかってしまっては、とても実用には耐えないと判断し、列形式データも逐次ディスクに保存する仕様としました。ただし、二重持ちと言ってもデータは圧縮しているため、ディスク使用量はそれなりに抑えられます。

具体的にどれくらい圧縮できるのでしょうか。

河場

データの種類によりますが、最大で 2 分の 1 くらいは圧縮可能です。圧縮によりディスク読み込み時の I/O 量も減らすことができます。列形式の共有バッファーには、一度データを載せるとデータを維持し続けるステーブルバッファー機能を搭載しています

 © 2016 Fujitsu Limited

が、それでも急激にデータ量が増えた場合などバッファーから溢れることがないとは言いきれません。その場合もディスクに列形式の圧縮データがあるので、安定した性能で使い続けることができます。

データを二重持ちするリスクを最小限に抑え、メリットへと転換しているのですね。これまで、インメモリカラムナ機能の仕組みについて伺ってきましたが、本機能は PostgreSQL にどのように実装されているのですか？

河場

インメモリカラムナ機能は、PostgreSQL のエクステンションとして実現しています。PostgreSQL の魅力のひとつはオープン性だと考えています。ですから、多くの PostgreSQL ユーザーに、富士通ならではの高性能・高信頼性なデータベース機能を提供できることを目指しています。

ということは、オープンソースの PostgreSQL との互換性は意識されましたか？

河場

それはかなり意識しました。インメモリカラムナ機能を実現するために、まったく新たなエンジンを開発・実装すると同時に、既存の PostgreSQL との互換性も絶対に維持しなくてはいけません。この 2 つの要件を両立させるために、かなり苦労を重ねました。

インメモリカラムナ機能の実現と PostgreSQL との互換性を維持するための苦労

どのような点で苦労されたのですか？

河場

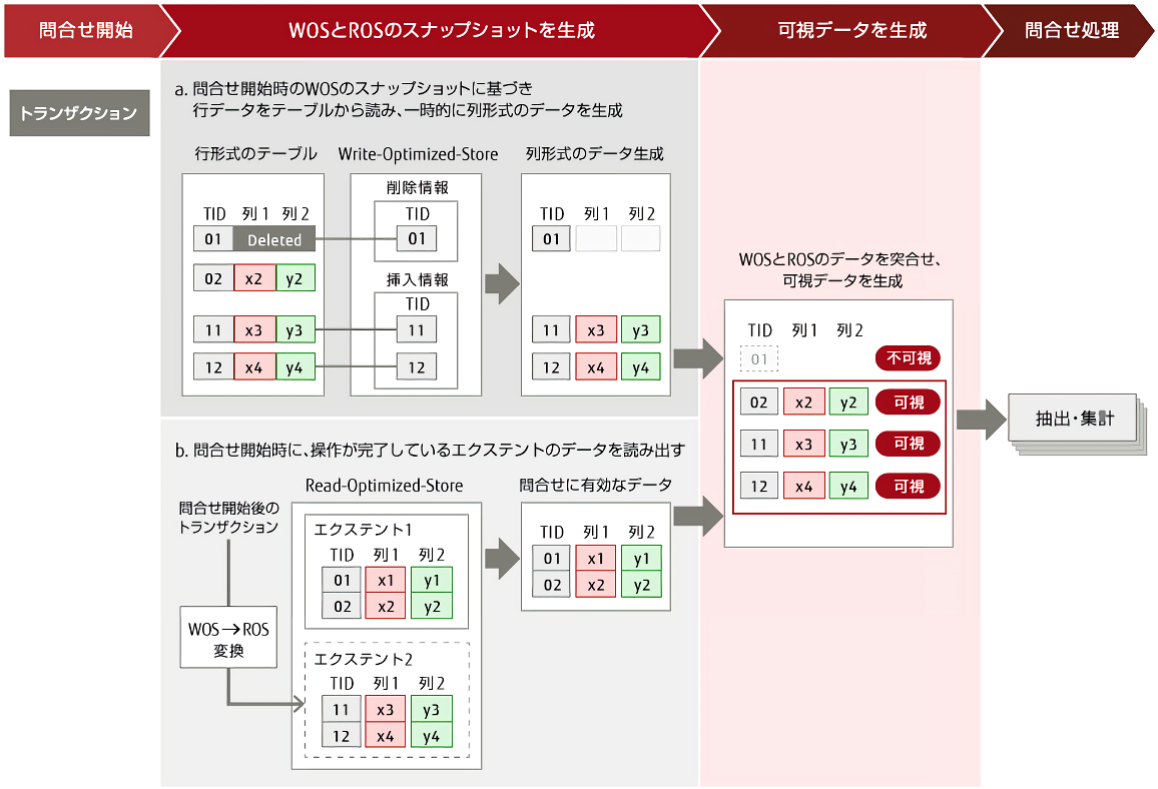
そうですね。例えば、エラーが発生した際にユーザーに表示するメッセージがあります。もともとの PostgreSQL と、新たに開発したエンジンとで異なっていては、ユーザーが混乱してしまいますからね。特に、苦労したのは、特定のトランザクションから見えていいデータと見えてはいけないデータをきちんと管理する「トランザクションの可視性」の互換性ですね。PostgreSQL は、テーブル内の各行に追加・削除を識別するトランザクション ID を持っていて、この情報を基に、テーブルをスキャンする際に「可視な行」だけを読みます。インメモリカラムナ機能でも、PostgreSQL と同様の一貫性を保たなければなりません。

インメモリカラムナでは WOS（行形式の追加・削除情報）と ROS（列形式のデータ）にデータが分かれていて、データ参照の際にはこの 2 つを突合せると伺いました。WOS と ROS を突合せる時に、どう一貫性を保つかということでしょうか。

河場

はい。WOS は更新トランザクションと同期しているので、更新の際に PostgreSQL のトランザクション ID の情報も記録しておくことで、どれが可視な行なのかがわかります。データ参照の際には、クエリ開始時点のスナップショットから、可視な行のデータを一時的に列形式に変換しておきます。一方、ROS には、不定期に WOS のデータが反映されます。一貫性を保つには、ROS に反映中のデータは、他のトランザクションから見えないようにしておく必要があります。そこで WOS から ROS への反映は、エクステントという単位で管理するようにしました。クエリ開始時には、データ反映が完了しているエクステントを判定し、そのスナップショットと、先ほどの一時的に列に変換したデータと突合せることで、一貫性を確保しました。こうした極めて細かな点までこだわり、互換性の維持に努めました。

 © 2016 Fujitsu Limited



図：WOS と ROS の突合せ時における一貫性を保つ仕組み

実際にデータベースを使うユーザーの立場に立つと、そうした細かなインターフェイス 1 つ 1 つの違いがアプリケーション開発の効率を大きく左右しますからね。

河場

そうですね。また、実装する際のアプローチを選ぶ上では、新たに実装する機能と既存機能との相性も考える必要がありました。同じ機能を実現するにも「性能面で有利なアプローチ」「変更量が少ないアプローチ」などさまざまなアプローチが考えられるのですが、それぞれ PostgreSQL との相性が異なります。そこで、個々の機能ごとになるべく多くの実装アプローチを挙げ、それらの中から PostgreSQL へ組み込むのに最も適したものを選んでいくという作業をひたすら繰り返しました。

やはりこれだけ先進的な機能を実現するとなると、一筋縄ではいかいのですね。ちなみに、今後はどのような形で PostgreSQL とかかわっていきたいとお考えですか？

河場

まずは、性能をより一層“尖らせて”いきたいですね。やはり、ある処理系において世界最速レベルを達成するというのは、技術者の 1 つの夢ですからね。また今後のデータベース技術の方向性として、データ分析の機能が徐々にデータベース製品に取り込まれていくのではないかと予想しています。メモリー容量をはじめとしたハードウェア性能が飛躍的に向上したことで、従来の OLTP に加えて OLAP も同じ箱で扱えるようになりました。そして今後もっとハードウェア性能が上がれば、OLAP だけに留まらずさらに広い領域の機能が同じ箱の中で実現できるようになることでしょう。

 © 2016 Fujitsu Limited

その 1 つが、データ分析機能だということですね。

河場

その通りです。データベースサーバーのメモリーがさらに増えれば、それまで別のサーバーで行われていたデータ分析処理もデータベース上で行えるようになるはずです。当面は、限られた分析処理だけに限られるでしょうが、将来的には機械学習や AI の処理まで PostgreSQL に取り込まれるようになるかもしれません。一技術者としては、そうした技術の研究や実用化にぜひ関わってみたいと考えています。

AI の処理が PostgreSQL に取り込まれる！それは夢のある話ですね。データベースの先行きが楽しみです。ありがとうございました。

2016 年 6 月 24 日

 © 2016 Fujitsu Limited