2C-03

PostgreSQL ベースの並列処理向けの共有メモリ機構の設計

宇治橋　善史[†]，中村　実[†]，田原　司睦[†]，橋田　拓志[†]，河場　基行[†]，原田　リリアン[†]
株式会社富士通研究所[†]

1. はじめに

大量のデータを蓄積し、高速に問い合わせに答えるシステムとして、古くからリレーショナルデータベースマネージメントシステム(RDBMS)が用いられている．RDBMS の用途としてはデータを高速に更新し，レコード単位の検索を高速に行なう OLTP 処理が一般的であったが，近年，データウェアハウス等で従来行われていた蓄積された大量のデータを読みだし統計処理を行うような OLAP と呼ばれる解析処理も RDBMS で実行し，更新データをリアルタイムに集計解析可能にする OLXP システムが注目されている[1]．

我々は PostgreSQL の OLXP システムを実現するためにカラムストアインデックス(CSI)を開発した[2][3]．本稿では，CSI の並列処理を実現するための基盤となっている共有メモリフレームワーク SMC を説明する．

2. PostgreSQL のメモリ管理機構と課題

CSI ではクエリ単位のプロセス並列処理を実現する．そのためには，各プロセス間のクエリ処理用データを共有し高速にデータの授受を行なうための共有メモリ機構が必要である． PostgreSQL が内部で使用しているメモリ管理システムは 4 種類ある．①ShmemInitStruct()によって提供される共有メモリ、②DB バッファ、③palloc()によって確保されるプロセスローカルメモリ、④PostgreSQL 9.4 から導入された Dynamic Shared Memory(DSM)である。このうち ShmemInitStruct()で確保される共有メモリはプロセス間で同一のアドレスに割り振られることが保証されているが，この領域は PostgreSQL のプロセス管理やシグナル処理等の基本機能のための限定用途領域であり，クエリ処理用データを格納するには適さない．

DSM はクエリ実行時に動的に確保できる共有メモリであるが，プロセス毎に別アドレスにマップされる可能性がある．プロセス毎にアドレスが異なると，直接ポインタを手繰る操作が不可能であるため，リンク構造を保ったまま，一旦ローカルプロセスにコピーする必要がある．Hash Aggregation 等の並列処理ではプロセス間で共有する Hash テーブルに頻繁な更新処理が発生する．その都度ローカルコピーを実行するとコピーオーバーヘッドによりクエリ性能が低下してしまう．これら以外のメモリ管理システムは共有メモリではない．よって，高速なクエリ単位並列処理を実現するためには，動的メモリ確保とプロセス間アドレス一致を兼ね備えた共有メモリ機構を新たに導入する必要がある．

また，PostgreSQL には MemoryContext というメモリ管理機構が備わっており，メモリ領域の確保・解放は MemoryContext に対する palloc()，pfree()をするという方法で行われる[4]． PostgreSQL で扱われるプランノードツリー等のオブジェクトは MemoryContext 上に構築されているので，既存コードを活用するためには，MemoryContext と等価な仕組みを持った共有メモリ機構が必要になる．

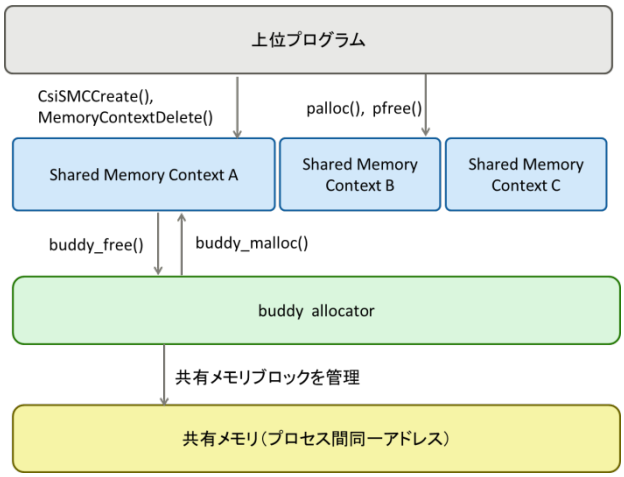

図１ SMC のソフトウェアスタック

3. 共有メモリフレームワーク（SMC）

我々は前節で述べた課題を解決するために，プロセス間同一アドレスを持ち，PostgreSQL

Design of shared memory mechanism for parallel processing on PostgreSQL

Ujibashi Yoshifumi[‡], Nakamura Minoru[‡], Tabaru Tsuguchika[‡], Hashida Takushi[‡], Kawaba Motoyuki[‡], Harada Lilian[‡]
Fujitsu Laboratories LTD.[‡]

の MemoryContext と共有の I/F を持つ共有メモリフレームワーク SMC(Shared Memory Context)を開発した．SMC のソフトウェアスタック構成を図１に記載する．SMC は Shared Memory Context，buddy allocator，共有メモリから構成される．

(ア) 共有メモリ（Linux の場合）

共有メモリの実体は/dev/shm のファイルである．各プロセスにおいてそのファイルに対して mmap()システムコールを実行することにより各プロセスのメモリアドレス空間に共有領域をマップする．mmap()の MAP_FIXED オプションにより各プロセス固定のアドレスにマップすることができる．共有メモリはクエリ開始時に作成し，クエリ終了時に破棄する．

(イ) buddy allocator

buddy allocator は共有メモリの割付領域を管理する．メモリ管理手法としてはバディブロックアロケータを採用している[5]．buddy allocator は Shared Memory Context からメモリ割付要求を受け付け，ブロック単位の領域を割付けて Shared Memory Context に返却する．メモリの割付をできるだけメモリ枠の低位アドレスから行なうことで，疎なファイルのなかでアクセスがあった箇所だけにメモリが割り当てられるようにする．

(ウ) Shared Memory Context

SMC は上位プログラムから Shared Memory Context 経由でアクセスされる．上位プログラムからみると，PostgreSQL の MemoryContext とほぼ同等の I/F を持っており，違いはメモリ確保・解放を共有メモリに対して行っていることだけである．この機能は，SMC 用の生成関数 CsiSMCCreate()によって生成された MemoryContext に対して，メモリ確保・解放関数 malloc()，free()を，buddy_allocator 用の buddy_malloc()，buddy_free()に挿し替えることによって実現している．

4. 並列処理機構における SMC

図 2 は CSI の並列処理機構を示している．クエリ要求はバックエンドプロセスが受け付ける．バックエンドプロセスは，SMC 作成とダイナミックバックグランドワーカ(DBW)プロセスを起動し，分割したクエリ処理を各 DBW に割り振ってクエリ並列処理を実行する．バックエンドと各 DBW 間で共有が必要なデータは SMC 上に格納し各プロセスで共有する．共有されるデータは，CSI の実行計画を格納するプランノードツリーや，DBW 間で共有する一時データ，各 DBW が出力する処理結果等である．ポインタ等のリンク構造を含むこれらのデータを，SMC で直接プロセス間共有することで，効率良く高速な並列クエリ実行を実現している．

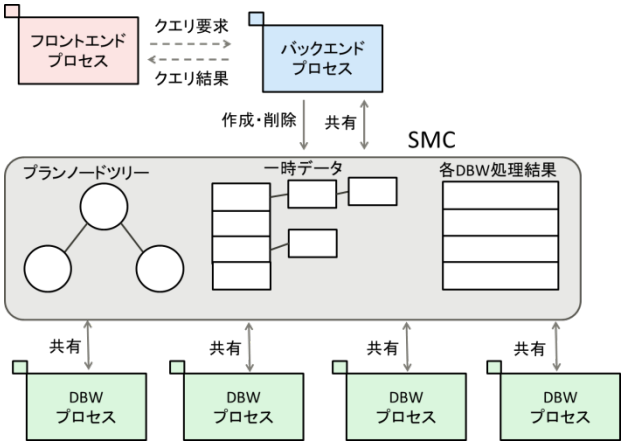

図 2 CSI の並列処理機構

5. まとめ

本稿では，我々が開発したカラムナインデックス（CSI）という PostgreSQL 拡張機能に備わる共有メモリ機構 SMC を説明した．SMC は CSI の機能に依存していないため，CSI と独立した単独機能としても利用可能である．今後は，CSI の共有メモリ機構としての SMC 評価を進めると共に，他機能における SMC 利用も検討する予定である．

[1] Hasso Plattner, The Impact of Columnar In-Memory Databases on Enterprise Systems, VLDB, 2014

[2] 田原 司睦他，カラムナデータをサポートするための PostgreSQL 拡張，第 77 回日本情報処理学会全国大会

[3] 橋田 拓志 他，PostgreSQL ベースのカラムナ機構へのトランザクション実現検討，第 77 回日本情報処理学会全国大会

[4] 井久保 寛明，メモリ管理，http://ikubo.x0.com/PostgreSQL/pdf/IK05_Memory_040702.pdf

[5]D.Knuth, *The Art of Computer Programming, Vol.1: Fundamental Algorisms*, p.443