

コンシューマ・システム論文

メモリ DB を活用したスーパー OLTP を実現する OMCS の PSA システムモデル

相澤 正俊^{1,a)} 富山 卓二² 斎川 幸貴³

受付日 2011年12月16日, 採録日 2012年4月13日

概要: OMCS のシステム事例としてモバイルコンピューティング時代のビルディングシステムや株価情報の配信などのそれまでのメインフレーム (MF) やオープンシステムでは不可能であった数万件/秒のスーパー OLTP の基盤として PSA (Parallel Stream Architecture) を開発した. PSA はメモリ処理を基幹システムの OLTP として実現した先駆的なシステムモデルでもある. PSA システムモデルのアーキテクチャをそれまでの OLTP との比較で説明し, 今後の大量イベント処理の基幹システムモデルとして重要になることを本論文で述べている.

キーワード: コンシューマサービス, OLTP, 並列処理, OMCS

PSA System Model of OMCS Realizing Super OLTP by Utilizing Memory DB

MASATOSHI AIWAZAWA^{1,a)} TAKUJI TOMIYAMA² KOKI SAIKAWA³

Received: December 16, 2011, Accepted: April 13, 2012

Abstract: This paper describes PSA (Parallel Stream Architecture) developed as a leading case of OMCS. PSA is adapted to billing system of mobile computing age or stock information searching, in which tens of thousand transactions capability is mandatory. Namely, PSA is a architecture for super OLTP that expand capability of Mainframe or previous Open System. PSA system model is a advanced system model which uses memory processing as the enterprise OLTP system.

Keywords: consumer service, OLTP, Parallel Processing, OMCS

1. はじめに

OMCS (Open Mission Critical Systems) とは, オープン製品やオープン技術を駆使して構築された大規模基幹システムと, それらを構築するための製品群および SI (System Integration) 技術のメソドロジ (方法論) の総称である. OMCS の歴史的背景, 発展の歴史の概説は参考文献 [1] に

詳しい.

1990 年代, パーソナルコンピュータとインターネットの普及により, ネットワークを介した情報共有や情報検索が一般化した, その後, 携帯電話からのインターネット接続が普及することで, その利用者数は飛躍的に伸張した. さらに, ここ数年のスマートフォン急増により, その利用形態が多様化, 従来の参照系主体の処理に加え, 商品購買にともなう決済処理や口座間の資金移動など, 高いサービス品質が必要とされる処理が増加傾向にある. 今後さらに, M2M による大量イベント処理やビッグデータ処理による多様なサービスが出現すると想定され, コンシューマ向けのサービスを提供するシステムには, 従来以上の高い性能, 高い拡張性, 高い信頼性が求められる.

NEC では 1990 年代から他社に先駆けて, オープン製品

¹ 国際社会経済研究所
Institute for International Socio-Economic Studies, Minato,
Tokyo 108-0073, Japan

² NEC システムテクノロジー株式会社

NEC System Technologies, Osaka 540-8551, Japan

³ 日本電気株式会社 OMCS・通信・メディアソリューション事業
本部

NEC Corporation, Minato, Tokyo 108-8001, Japan

^{a)} m-aizawa@bc.jp.nec.com

やオープン技術を駆使した大規模基幹システムの構築プロジェクトに着手し、これまで多様な基幹システム構築プロジェクトの実践を通し、システムインテグレーション技術 (SI 技術) を確立してきた。その成果の代表的事例は、NTT ドコモの i モードシステムである。数千万の携帯電話からのメールやインターネットアクセスを可能とするために、1 千台を超えるサーバやネットワーク機器が稼働している。この巨大システムに万一大きな障害が発生した場合の社会的インパクトは容易に想像できるであろう。こういった大規模基幹システムを安全かつ確実に構築することを支えたのが、OMCS の SI 技術であり、この確立された OMCS の SI 技術は、今後ますます多様化するコンシューマ向けサービスを提供するシステム構築に必要となる技術である。

OMCS を支える中核テクノロジーの 1 つに、Mission Critical な特性を持つ「システム部品」の組合せからなるシステムモデルがあり、それは参考文献 [2] に詳しい。

携帯電話の普及にともない、電話料金のリアルタイムな集計が求められるようになってきた。数千万加入の携帯電話の通話にともない刻々と生成される課金情報を瞬時に、かつ、漏れなく集計するためには、新しい処理方式が必要となった。課金システムの障害は大きな社会的影響を与えることは明らかであり、新しい処理方式の確立とともに、OMCS の SI 技術を駆使して実システムを構築した。これは新しいシステムモデルとして今後も展開が期待されるものである。

本論文では、多量に存在するイベント発生源から随時発生する大量なイベントを、リアルタイム性を損なうことなく、かつ、漏れ・重複なく処理するための技術につき、説明する。

2. 従来技術の分析

大量に発生するイベントデータを取り扱うリアルタイム処理は、様々な研究や製品化が行われてきた。

2003 年にスタンフォード大学で発表された STREAM [7] は、リアルタイムに発生する大量データ (以降、データストリームと称す) に対して検索の問合せを行うものであった。データストリームを検索するために、すべてのイベントデータを蓄積してから DB で検索する方式ではなく、時系列に到達するイベントデータをリアルタイムに分析し分類する方式である。

その後、データストリームを分析・検索する処理について様々なタイプの研究活動や製品化が行われた [8], [9] が、これらの研究および製品は、リアルタイムな検索や分析を行うものであり、データストリームのイベントデータを入力として漏れ・重複のない処理、すなわち、トランザクション処理を行うものではなく、課金処理には向いていない。

一方、課金処理に古くから使われている方式は、バッチ

処理や OLTP (On Line Transaction Processing) であるが、バッチ処理はリアルタイム性に欠けるので、比較は OLTP が中心になろう。

既存の OLTP モニタと DBMS (Data Base Management System) との組合せによる OLTP には、商用システムの観点からはシステム上の限界点がある。クライアントからの要求を処理するために、AP サーバを多重化して並列実行するが、DB サーバへのアクセスが集中し DB サーバがボトルネックとなる。DB サーバのボトルネックを解消しようとする、システム構築のコストがかなり大きなものになってしまう。これが商用システムとしての限界を決めることになる。そのため、都市銀行の勘定系システム、航空券予約システムなど、非常に負荷の高いシステムでも、OLTP のトランザクション性能の最大は 3,000 件/秒程度となっている。

ところが、携帯電話の料金計算をリアルタイムに処理する場合や株価情報をリアルタイムに配信する場合では、数万件/秒という性能が必要とされる。

この数万件/秒というトランザクション処理をこなす OLTP をスーパー OLTP と呼び、スーパー OLTP を実現するアーキテクチャが PSA (Parallel Stream Architecture) であり、それをシステムモデル化 [1], [2] したものが、PSA システムモデルである。

3. 要件の整理

スーパー OLTP として処理すべきデータは、「イベント」として大量に発生し、データセンタ側にリアルタイム処理すべきデータとして大量に集まる。1 問 1 答型のトランザクション処理とは異なり、クライアントから一方向的に大量のイベントをセンタ側に送信するタイプ (いわゆる、集信型) のトランザクション処理である。

(1) 業務モデル

以下のような業務モデルを想定する。

- イベントの発生源が数多く存在し、その発生源から多くのイベントが発生する。
- イベントの発生源ごとの集計のほかに、イベントの発生源を何らかの形でグルーピングした単位での集計が必要な業務である。たとえば、以下のような例が考えられる。
- 携帯電話ごとに通話記録から料金を集計し、家族や法人などの単位での集計を行って割引計算をし、請求者ごとに集計して請求金額を計算する必要がある業務

なお、この業務モデルは PSA の本質を理解しやすいように、単純化したものを提示している。PSA に適合する業務特性については、本論文の最後で触れる。

(2) システム要件

以下のシステム要件を想定する。

- 性能：イベント処理性能：数万件/秒
- 即時性：ほぼリアルタイムに集計が行われること
- 高可用性：サーバの障害などが発生しても、集計に抜けや重複がないこと
- 拡張性：トラフィックの伸びに対し、アプリケーションの修正なく、システムを拡張できること
- 経済性：商用システムとして、現実的な費用で実現できること

4. PSA (Parallel Stream Architecture)

以下、PSAを実現するために、考慮しなければならないこと（課題）、課題解決のための基本的アイデア、ならびに実装について説明する。

4.1 考慮すべきこと（課題）

- ① イベントの処理に必要となる、アプリケーションロジック以外の部分のオーバーヘッドをなるべく少なくする必要がある。

従来の OLTP の処理は、即時性に優れるが、基本的に 1 つのイベントごとに処理を行うので、イベントの入出力処理やそのコミット処理を中心とする、アプリケーションロジック以外の部分のオーバーヘッドが大きくなる。従来のバッチ処理は、複数のイベントを 1 度に処理するので、アプリケーションロジック以外の部分の重さは、相対的に低くなるが、バッチ処理はリアルタイム性に欠ける。したがって、従来の OLTP とバッチの中間に位置する処理方式が必要である。図 1 は、PSA が目指そうとしている処理パターンのイメージである。

- ② 単純なアプリケーションロジックに見合った、データアクセス手法が必要である。

従来の OLTP と DBMS で処理を実現する場合には、SQL が利用されるが、単純な集計処理の場合、SQL 自体が消費する CPU や DB サーバへの通信などのオーバーヘッドが大きい。したがって、より軽いデータアクセス手法が必要である。

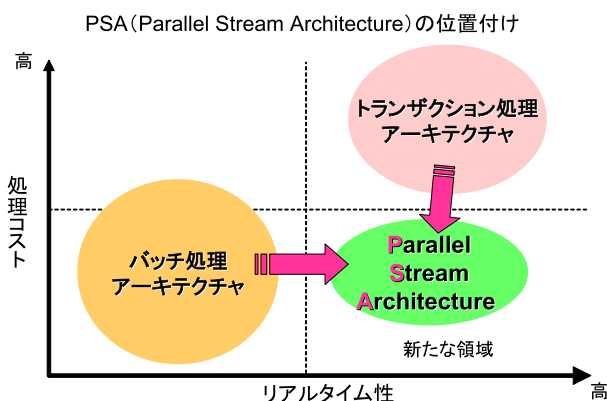


図 1 PSA の特性

Fig. 1 Characteristics of PSA.

- ③ アプリケーションの動作にともなう外部記憶装置への I/O 回数を少なくする。

大量のイベントを処理するためには、処理の経過時間を短くすることが重要であり、外部記憶への I/O 回数を削減するという考え方は、しごく自然な考え方である。メモリ DB を用いて外部記憶への I/O を極力抑える方式が必要である。

- ④ あるアプリケーションの実行が、他のアプリケーションの実行により占有されている資源の解放を待ち合わせるなどの「待ち状態」を作らない。

図 2 は、従来の OLTP システムでの資源の競合をイメージした図である。

たとえば、あるプロセス A で動作するアプリケーションが、別のプロセス B で動作するアプリケーションにより占有されている資源（DB、バッファ、ロックなど）を待ち合わせてしまうと、プロセス A が占有されたままとなり、全体としてのスループットが上がらなくなる。

一般に DBMS を利用すると、テーブルを別にしたとしても、DBMS 中にある共有バッファやロックがネックとなり、スループットが上がらなくなる。したがって、共有資源を持たない DBMS 相当の機能が必要となる。上記②と合わせると、新たな DBMS の作成が必要になることを示唆している。

また、アプリケーションが動作するプロセス（ないしスレッド）にデータを括り付け、あるデータはあるプロセス（ないしスレッド）からしかアクセスできない状況を作り出す必要がある。

- ⑤ 業務特性から生じる注意点：処理の分割が必要になる
イベント発生源が多数存在し、イベント発生源ごとに集計処理が必要であることから、個々のイベント発生源を、アプリケーションが動作するプロセス（ないしスレッド）に割り当て、プロセス（ないしスレッド）ごとにいくつか

プロセス間でDBMSの共有資源競合発生

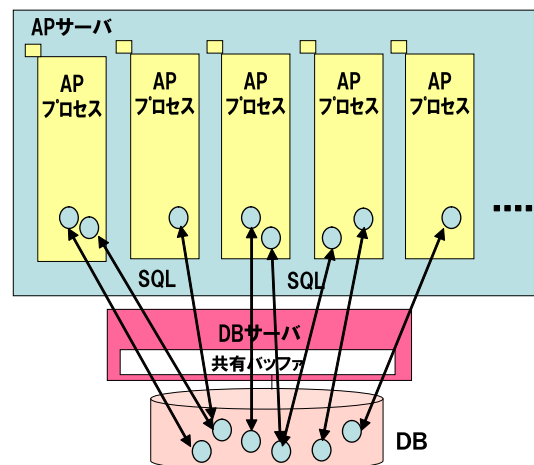


図 2 OLTP での資源の競合イメージ

Fig. 2 Resource conflicts in OLTP.

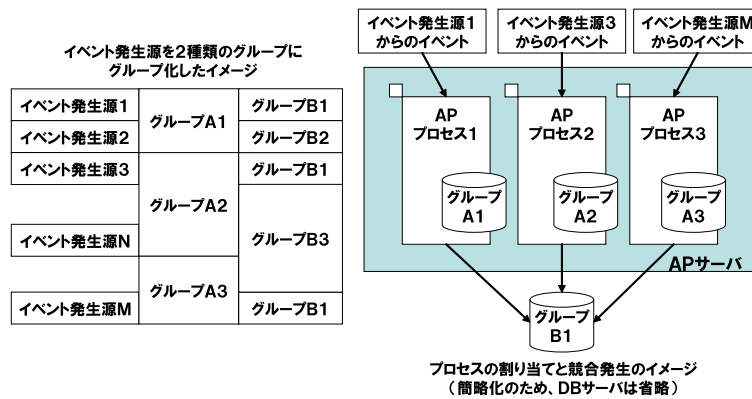


図 3 待ちの発生

Fig. 3 Resource conflicts.

のイベント発生源の集計を行うと考えるのは自然な考え方である。

一方、イベント発生源は、何らかのグルーピングが行われ、それを単位に集計も行わなければならない。またさらに別のグルーピングが行われ、それを単位に集計も行わなければならない。この2つ目、3つ目の集計を同じプロセス（ないしスレッド）上で行うとすると、プロセス（ないしスレッド）で共通のデータを更新することになり、「待ち状態」が発生してしまう。

図 3 は、その様子を説明するものである。なお、ここでは簡略化のために、DB サーバは記載していない。プロセスと DB との関係のみに着目すれば十分である。

イベント発生源 1, 2 からのイベントは、グループ A1 として集計され、イベント発生源 3, N からのイベントはグループ A2 として、イベント発生源 M からのイベントはグループ A3 として集計されるとする。同時に、イベント発生源 1, 3, M は、グループ B1 としての集計も必要であるとする。

AP プロセス 1, 2, 3 は、それぞれグループ A1, A2, A3 に属するイベントを処理することにした場合、イベント発生源 1, 3, M からのイベントは、それぞれ、プロセス 1, 2, 3 で個別に処理されるが、同時にグループ B1 としての集計を行おうとすると、プロセス 1, 2, 3 で競合による待ちが発生する。

こういった競合を避けるためには、AP プロセス 1, 2, 3 では、グループ A1, A2, A3 の処理をいったん完了し、グループ B1 の処理を専門に行う別の AP プロセスに処理を継続させるような仕組みが必要になる。図 4 は、そのイメージである。すなわち、従来の OLTP では通常 1 回のトランザクション実行で完了するようにアプリケーションが作成されるような処理を、資源の競合による待ちの発生を極力防ぐために、複数のステップに分けて実行させるような仕組みを作る必要があるということである。

この複数のステップに分けることを、処理を「処理ステージに分解する」と呼ぶことにする。

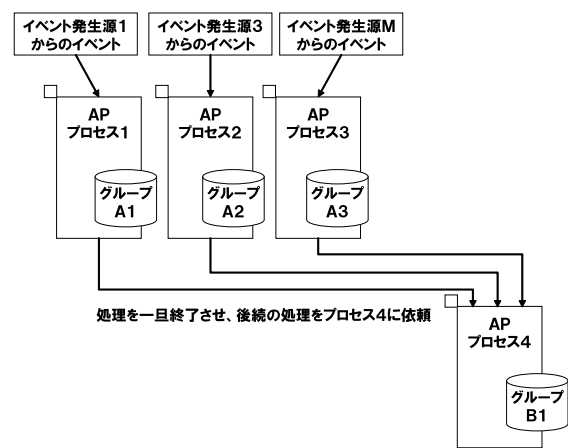


図 4 後続の処理を別プロセスに依頼

Fig. 4 Split the processing of subsequent.

4.2 課題解決のための基本的なアイディア

- ① アプリケーションはスレッドで動作させる。大量のイベントを処理するために大量のプロセスないしスレッドが必要になるが、OMCS として構築を経験した i モードシステム [1] で、数万のスレッドを制御する手法が確立されていたためである。
- ② 個々のスレッドに、そのスレッドで処理されるべきデータを振り付け、そのデータをアクセスすべきイベントは、そのスレッドで動作させることにする。このスレッド上のアプリケーションとデータのセットを「ストリーム」と呼ぶ
- ③ ストリームで処理されるイベントは、複数イベントを一括して処理する。イベントの発生頻度を考慮し、処理の即時性が損なわれない程度に複数イベントをまとめてストリームに渡し、処理を行わせる。複数件を一括して処理するとすれば、アプリケーションロジック以外のオーバーヘッドは、相対的に削減することができる。バッチ処理の特徴を生かすのが狙いである。
- ④ ストリームに渡される複数のイベントを格納したものを「コンテナ」と呼ぶ。

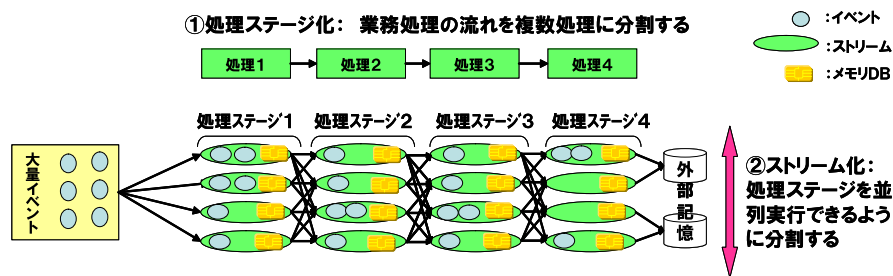


図 5 PSA の処理の流れ

Fig. 5 Processing flow of PSA.

- ⑤ 1つのストリーム A で処理されたイベントは、次の処理を行うべきストリームに渡される。ストリーム A で一括処理されたイベントのうち、いくつかは、後続のストリーム B1 に渡され、いくつかは、後続のストリーム B2 に渡される。グループ B1 に属するイベントを処理するのに必要なデータは、ストリーム B1 が動作するスレッドに括り付けておき、グループ B2 に属するイベントを処理するのに必要なデータは、ストリーム B2 が動作するスレッドに括り付けておく。
- ⑥ ストリーム B1 に対しても、イベントの発生頻度を考慮しつつ、処理の即時性が損なわれない程度に複数イベントがコンテナとしてまとめられて渡され、処理される。ストリーム間でコンテナを受け渡す仕組みを「コンテナ転送機構」と呼ぶ。
- ⑦ これを繰り返すことにより、複数のグループ化を行った集計を順次行ってゆくことができる。
- ⑧ この方式では、ストリーム A で行われた処理と、その後続として行われるストリーム Bx の処理は、一貫して行われる必要があり、ストリーム Bx に渡されるべきイベントが紛失することや重複して処理されることは許されない。したがって、コンテナ転送機構は、外部記憶を用いて、紛失や重複を防ぐ仕組みを持たせる。
- ⑨ ストリームで処理されるデータは、メモリ上に展開される。これをメモリテーブルと称する。このデータは該当ストリーム（が動作するスレッド）で占有されアクセスされるだけでなく、他のストリームとの間で、共有バッファやロックなどの共有資源を持たず、また、従来の DBMS に比べ、アクセスオーバーヘッドの少ないものを用意する。
- ⑩ サーバの障害などに備え、メモリテーブルを復旧する仕組みが用意される。同時にコンテナの内容も矛盾なく復旧される仕組みが用意される。

以上が基本的アイデアであり、それをさらにまとめてみると、以下ようになる。

- 4回の集計処理が必要だとして、4つの集計処理を、それぞれバラバラにし（各々の処理を処理ステージと称する）、個々の処理ステージごとに、ストリームを複

数用意して、ストリームどうしがお互いに干渉しあわないように、並行動作させる。

- ストリームには複数件のイベントを一括して渡し、処理させることで、オーバーヘッドの削減を図る。
- 従来の DBMS に比べ、アクセスオーバーヘッドの少ないメモリテーブルを用意する。
- ストリーム間のイベント受け渡しの仕組みを用意し、メモリテーブルとの整合性も担保する。

4.3 実装

業務処理を多段の処理ステージに分割し、さらに処理ステージの中を並列実行できるように分割し（分割した実行単位をストリームと定義）、メモリ DB を使用して高速な並列処理を実現する。このアーキテクチャを PSA と呼び、具体的な処理の流れの概要を図 5 に示す。前節で説明した 4 つの集計処理が 4 つのステージになっている。

データ処理はメモリテーブル上で行われるが、最終的な処理結果を外部記憶に書き出すことも可能である。たとえば、メモリテーブル上で料金計算を行った結果を最終的に外部記憶に書き出し、それをオンラインで参照するような使い方を想定している。

PSA を実現するために、超並列処理エンジン（ミドルウェア）を製品化した。

ストリームの実行制御、イベントスケジュールのためには、PSM^{*1}（Parallel Stream Monitor）を開発した。PSM はストリームへのイベントの配信、ストリーム上のコミット制御、後段のストリームへのイベント配信を行う。

ストリームはスレッド上で動作する（1 スレッドに複数のストリームを定義可能な実装となっている）。PSM はイベントが入力されると、アプリケーションにイベントを渡し、アプリケーションはメモリ DB にアクセスして業務処理を実行した後、後段ストリームの宛先を指定してイベント出力を PSM に依頼する。

メモリ DB として TAM^{*2}（Table Access Method）を開発した。TAM はインデックスによる検索更新が可能であり、アクセス用の各種 API を持つ。メモリ DB のメモリ

^{*1} 製品の正式名称は、WebOTX Parallel Stream Monitor である。

^{*2} 製品の正式名称は、InfoFrame Table Access Method である。

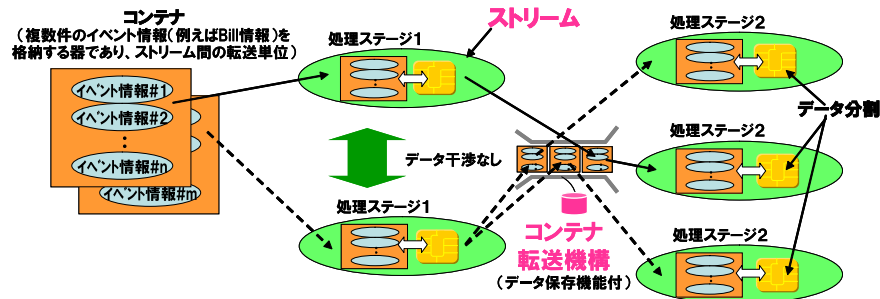


図 6 PSA の構成要素

Fig. 6 Components of PSA.

テーブルは、システム起動時、外部記憶装置から読み出され、システム終了時、外部記憶装置に書き出される。メモリテーブルの内容をコミットする際にシステム障害時の復旧用ログが採取されること以外、メモリテーブルへのアクセスを契機として外部記憶装置へのアクセスが発生することはない。

PSA の構成要素は図 6 に示すとおりであり、具体的には以下の構成となる。

① ストリームとデータ分割

1つの処理ステージは、複数のストリームを多重実行させる。高スループットを実現させるためには、ストリーム間で競合が発生しないことが必要であり、メモリ DB の 1 つのメモリテーブルは、特定の 1 つのストリームからのアクセスしか許さないことを基本とする。複数のストリームから参照されるような共有テーブルは実装可能であるが、更新アクセスは特定のスレッドからのみ可能となっている。

各ストリームが並列実行できるように、各ストリームが扱う更新用データ（メモリ DB のデータ）はストリーム間で重複しないようにデータ分割して割り当てる。データ分割の方法は、業務処理依存であり、メモリテーブルの主キーに対して、レンジ分割やハッシュやキー値の一部（下 N 桁）などを使って分割する。この分割によって、各ストリームへのイベントの行き先が決定されるため、ある特定のストリームにイベントが偏らないように、各ストリームへのイベント流入量が均等になるようなデータ分割を行うよう、業務設計することが望ましい。このようなデータ分割を前提としている点が PSA の大きな特徴である。

② コンテナ転送機構

複数のイベント情報はコンテナに格納され、ストリーム間で転送される。コンテナに格納されるイベントの数は、リアルタイム性を損ねない程度に事前設計される。コンテナにより渡された複数のイベントは、ストリームで一括処理される。

コンテナがサーバ間をまたがる場合はネットワーク転送となるが、システム障害時などのコンテナの消失を防ぐために、メモリテーブルの内容をコミットする際に、コンテナを外部記憶装置に格納する。

4.4 技術的工夫点

(1) トランザクション処理の実現 [特許 4232109 号] [11]

PSA は入力イベントを読み込んで、アプリケーションを実行し、アプリケーションはメモリ DB の更新とイベントを出力するといったトランザクション処理が可能である。コミット対象となるリソースは入力イベント、メモリ DB、出力イベントとなる。障害時には、これらのリソースの状態をロールバックして再実行することができる。

また、PSA は複数の入力イベントに対して一括してトランザクション処理することが可能である。一括処理によりオーバーヘッドの削減が可能である。

(2) ストリーム選択方式

後続のストリームを選択するのは、アプリケーションの責任である。ストリームに渡されるイベントには、イベント発生源を識別する情報が入っている。各ストリームには、イベント発生源と後続の処理を行うストリームの対応関係をメモリテーブルとして保持しておくのが普通である。このメモリテーブルを宛先用メモリテーブルと呼ぶことにする。

ストリーム上のアプリケーションは、イベントの発生源を識別する情報と宛先用メモリテーブルを用いて、後続の処理を行うストリームを決定し、それを PSM に通知する。こういった構造は、宛先用メモリテーブルを入れ替えるだけで、後続の処理を行うストリームを変更できることを意味している。アプリケーションの修正をとまわず、処理ステージの構造を変更することができる。たとえば、携帯電話の契約条件が変更になることで、ある処理ステージの集計が不要になるようなケースを想定する。あるイベント発生源からのイベントについて 2 番目の処理ステージが不要になった場合には、宛先用メモリテーブルを変更し、後続のストリームを 3 番目の処理ステージのストリームに変更すればよい。新しい処理ステージの追加が必要になった場合も、同様に宛先用メモリテーブルの修正で対応可能である。

加えてこの構造は、ストリームを増やす場合にも有効である。ある処理ステージの 1 つのストリームを分割する場合や、新たなイベント発生源が増えたので、別のストリー

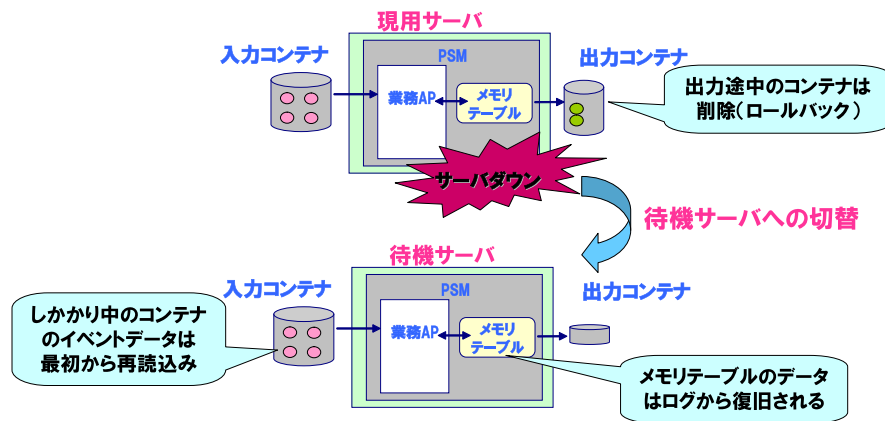


図 7 サーバ切替え動作

Fig. 7 Server switching operations.

ムを新たに作る場合にも、宛先用メモリテーブルを置き換えることで対応可能である。

一方、最初のストリームの決定は PSM が行う。この決定は、イベント発生元の識別子の一部をハッシュ化するなどの手法であらかじめ定められる。最初の処理ステージのストリームのアプリケーションは、自分に振り分けられるイベント発生元のみを処理すればよく、たとえば上記の後続のストリームを決定するための宛先用メモリテーブルも、自身が処理するイベント発生元のテーブルを持っていれば十分である。

(3) スレッド数やストリーム数の決定方法 (ストリームの再配置)

1つのストリームは1つのスレッドで実行される。1つのスレッドでは複数のストリームを実行させるようにできる。目標とする性能 (たとえば数万件/秒) を満足するようなストリームの多重度を決定する必要がある。まず1つのステージのストリームの処理性能を求め、目指す目標を満足するためのスレッド数を (ある程度の余裕を見て) 計算する。

問題は、スレッドになるべく均等にイベント処理を行わせるための工夫である。イベントの発生を完全に予測することは困難である。イベントを処理するストリーム (ここには処理すべきデータが括り付けられている) の分割があまり適当でないと、あるストリームに処理が集中してしまい、結局1つのスレッドでは処理しきれないことが発生する可能性がある。したがって、ストリームの分割は、あまり大きくせず、1つのスレッドに複数のストリームを割り当てるように設計する。個々のストリームには、なるべく均等にイベントが発生するように設計を行うことが望ましいが、実際にシステムを稼働させてみると、イベントの発生に偏りが生じ、あるストリームに負荷が集中することも考えられる。こういったことに対処するために、ストリームのスレッドへの割当てを変更できる機能が搭載されている。これにより、負荷の高いストリームと負荷の低いスト

リームと組み合わせるなど、動的な対応が可能である。

なお、この機能は負荷の増大に対する拡張性の担保としても利用されている。それについては、5.4 節で述べる。

(4) 高可用性の実現

トランザクションのしかかり中に (1 コミット単位となる複数のイベントを処理中に) サーバダウンが発生した場合、サーバ切替え後に、処理途中であった入力コンテナの復旧や出力途中のコンテナの削除を行い、旧現用系で出力されていたログを用いて、待機系 (新現用系) でメモリテーブルが復旧され、再度、入力コンテナからイベントデータを読み込んで処理を再開する (図 7 参照)。

これらの処理は、PSM と TAM が連携して行うので、イベントデータの紛失、重複したイベント処理や重複したイベント送信は行われない。また、PSM ではコンテナの中のイベントの処理順序は変わることはないため、イベントの追い越しも発生しない。

この方式の場合には、待機系 (新現用系) でのメモリテーブルの復旧は、系の切替え時に行われるので、系の切替えが発生するまで待機系で事前に処理を行っておく必要がなく、たとえば現用系 4 台に対し待機系 2 台といった柔軟な構成をとることが可能であり、システム構築コストを抑えることができるが、メモリテーブルの復旧には時間がかかる。

メモリテーブルの復旧の高速化を目的として、現用系サーバのメモリテーブルのデータを待機系サーバにレプリケーションしておくオプションも用意されている [特許第 4277873 号] [12] (図 8 参照)。

レプリケーションは通信によって行い、複数の LAN を束ねて送信することが可能である。LAN 障害についても考慮してあり、1つの LAN に障害が発生しても残りの LAN により通信を行い、レプリケーションを継続することができる。この方式の場合には、待機系にあらかじめメモリテーブルのデータを送っておくことになるので、1つの現用系に対して必ず1つの待機系を必要とする。このため、

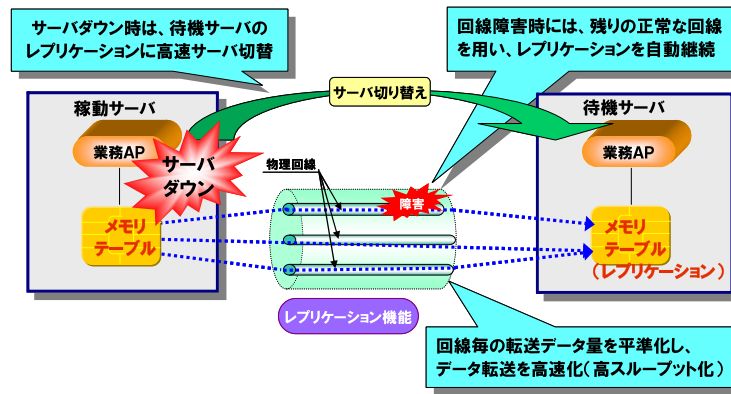


図 8 高速サーバ切替え

Fig. 8 High-speed server switching operations.

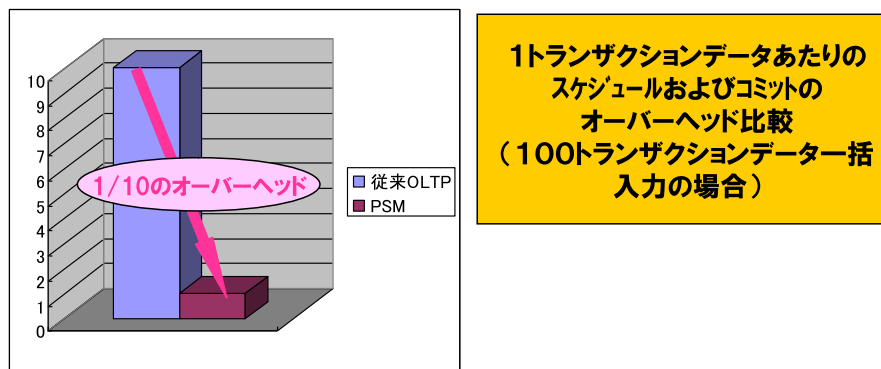


図 9 PSM によるオーバーヘッド削減

Fig. 9 Reduction of overhead.

メモリテーブルの復旧の時間は短縮^{*3}されるが、システム構築コストが高くなることには注意が必要である。

5. 評価

5.1 トランザクションオーバーヘッドの削減効果

PSA では、複数のイベントを一括で処理することにより、コミット処理やデータ入出力のオーバーヘッドを相対的に小さくすることを目指した。実測では、100 件を個別に処理するケースと 100 件を一括処理するケースでは、コミットやデータ入出力のオーバーヘッドが 1/10 まで低減することを確認した (図 9)。

図 9 の中の「従来 OLTP」とは、TUXEDO 8J と Oracle 9i で、100 件のイベントを 1 件ずつ処理した場合の 100 件分のオーバーヘッドの合計であり、「PSM」とは、PSM と TAM で 100 件のイベントを一括して処理した場合のオーバーヘッドである。

この測定では、従来 OLTP と PSM とのミドルウェア性能の比較であり、OLTP と PSM 上でのアプリケーション処理の差分 (SQL とメモリアクセスの差分) は含まれていない。

^{*3} システム障害に巻き込まれたたかだか数件のレプリケーションをログから復旧する程度。これは通常数秒であるが、実際のシステム切替えには、他の処理も含めて 20~30 秒程度かかる。

100 件の一括処理により、1/100 に改善されないことに関して説明する。1 件のイベントの処理に要するオーバーヘッドの内訳を調べると、100 件の一括処理により 1/100 になる部分と、一括処理しようとしまいと変わらない部分がある。たとえば、イベントの入力や出力のために実行される命令の回数は 1/100 になるが、イベントデータの転送自体に必要なオーバーヘッドは、個別に読み込んでも一括で読み込んでも総量は変わらない。一括処理で削減できる処理と削減できない部分については

A: 100 件の一括処理でも変わらない部分

B: 100 件の一括処理で、1/100 になる処理部分

として、 $B \equiv 10A$ であることが計測できている。この場合、100 件を個別に処理する場合のオーバーヘッド X は

$$X = 100 (A + B)$$

100 件を一括して処理する場合のオーバーヘッド Y は

$$Y = 100A + B$$

であり、 $B \equiv 10A$ のときには、 $X \equiv 10Y$ となる。したがって、この計測結果は妥当である。

さらに、1,000 件を個別処理する場合のオーバーヘッドは、

$$X = 1000 (A + B) \equiv 1000 * 11A = 11000A$$

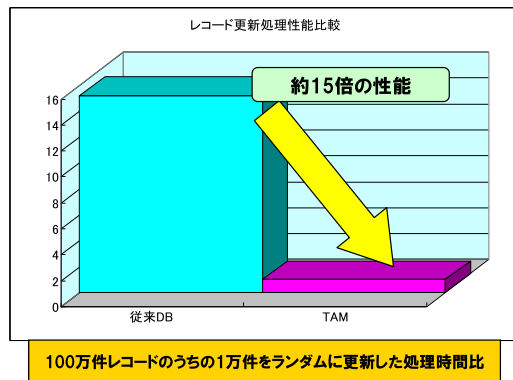


図 10 TAM の高速性能
Fig. 10 High-speed performance of TAM.

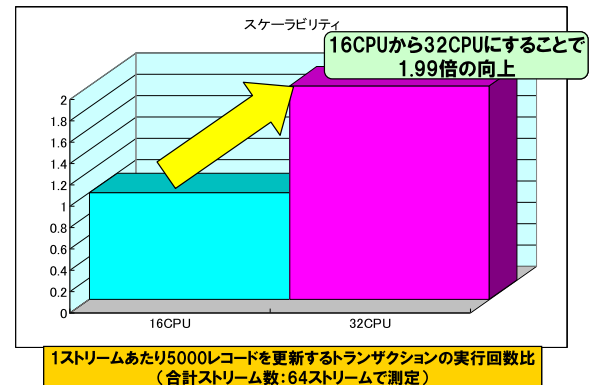


図 11 TAM のスケーラビリティ
Fig. 11 Scalability of TAM.

であり、1,000 件を一括処理する場合のオーバーヘッドは、 $Y = 1000A + B \approx 1010A$ となり、やはり、ほぼ 1/10 である。

■ 図 9 測定環境

マシン：4CPU (Intel(R) Itanium 2 processors
[1.5 GHz])
メモリ：16 GB
OS：HP-UX 11iv2
OLTP：TUXEDO 8J
DB：Oracle 9i

・測定条件

入出力データ長：256 byte
レコード長：256 byte
キー長：8 byte

5.2 メモリテーブル性能

PSA では、通常の DBMS よりも軽いアクセスメソッドの実現を目指した。TAM は DBMS の 15 倍スループットが良く、大幅な性能差となった (図 10)。

図 10 の中の「従来 DB」とは、Oracle9i を用いて、100 万件レコードのうちの 1 万件をランダムに更新した場合の処理時間であり、「TAM」とあるのは、TAM を用いて同一条件の更新をした場合の処理時間である。

■ 図 10 測定環境

マシン：HP superdome (Itanium2 1.5 GHz)
メモリ：512 GB
OS：HP-UX 11i v2 (11.23)
従来 DB：Oracle9i

・測定条件

レコード長：128 バイト
レコード件数：1,000 万件
インデックス長：12 バイト

また、TAM は共有の内部バッファ、ロックなどの共有リソースを持たず、あるストリームの動作が別のストリームの動作に影響しないことを目指した。共有リソースを持

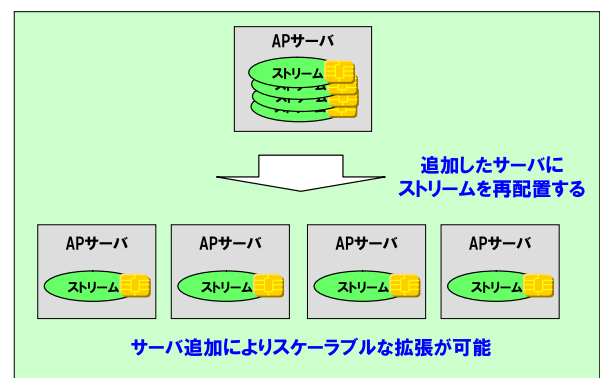


図 12 高拡張性
Fig. 12 High scalability of PSA.

たず排他制御が不要な構造にしたため、実行多重度を上げるとスケーラブルに性能が向上することを確認できた。実測では、CPU 数を 2 倍に増加したときの TAM へのアクセス回数は、1.99 倍に増加した (図 11)。

■ 図 11 測定環境

マシン：HP superdome (Itanium2 1.5 GHz)
メモリ：512 GB
OS：HP-UX 11i v2 (11.23)
・測定条件
レコード長：128 バイト
レコード件数：1,000 万件
インデックス長：12 バイト

5.3 高可用性

PSA では、サーバの障害などが発生しても、処理すべきイベントの抜けや重複がないことを目指した。

4.4 節 (5) で述べたように、TAM (メモリ DB) 上の処理データは、ディスクへの書き込みと別サーバへの複製を行うことによって、サーバ障害後のメモリテーブル復旧を行うことができる。

これにより、高可用性は満たされた。

PSA適用事例：リアルタイム料金システム

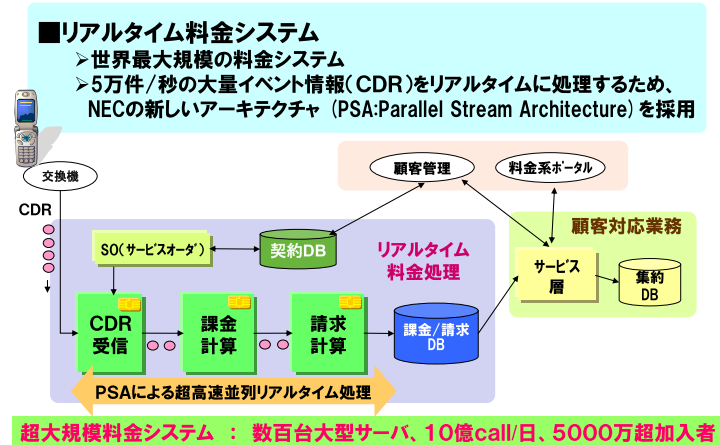


図 13 ドコモ様の料金システム事例

Fig. 13 Billing system of NTT DoCoMo.

5.4 高拡張性

PSA では、トラフィックの伸びに対し、アプリケーションの修正なしに、システムを拡張できることを目指した。

PSA のストリームが、互いに独立であることと、コンテンツ転送機構がサーバ間のコンテンツ転送をサポートしていることから、容易な拡張が可能である。すなわち、PSA にてスループット増加となったときには、サーバ追加を行いストリームの再配置を PSM に指示^{*4}してストリームを別のサーバに移動させることにより、スケラブルに処理能力を拡張することができる。図 12 はそのイメージ図である。これで高拡張性は満たされた。

5.5 その他の要件の評価

PSA で目指したその他の要件は、数万件/秒の性能と、商用システムとして現実的な費用で実現できる経済性である。また、各種性能改善効果の十分性も示す必要があるが、これらについては、適用事例で評価する。

6. PSA で実現したシステム事例

NEC は、PSA の製品化を行い (2007 年 9 月 14 日発表)、商用システムへの適用を行った。

6.1 ドコモ MoBills

PSA はドコモの料金システム (MoBills) をターゲットとして、2003 年 12 月から検討を開始し、2007 年に世界初のリアルタイム課金を実現した。携帯電話の通信明細データ (CDR) を受信して、その CDR をもとに通話料や割引額、請求額をリアルタイムに計算する。CDR は 10 億呼/日、ピーク性能は 5 万件/秒となった [3]。

ドコモ料金システムは、5,000 万人を超える利用者の料金

処理を行う、社会的に重要な大規模システムであり、PSA 上に大規模なアプリケーションが開発された。このため、OMCS^{*5} [1] としてのシステム要件が求められた。すなわち、メモリ処理による高性能に加えて、サーバ故障時にデータの紛失や重複した処理がなく、かつ、24 時間 365 日連続運転するための高可用性・高運用性、負荷の増大に柔軟に対応できる高拡張性が求められた。

課金計算と請求計算は処理を分割しており、図 13 のようなイメージとなる。課金計算では、個々のユーザの使用量をメモリ DB に配置し累計処理を行い、各種割引計算を行う。ストリームにはユーザが均等になうように割り当てる。

請求計算では、各ユーザの請求先となる請求者 ID ごとにデータを分割して、ストリームに割り当てる。各ユーザの課金情報をもとにして請求明細を作成する。

外部記憶上の DB には、課金計算結果、請求計算結果 (請求明細) の情報が格納される。これらはサービス層からオンラインでアクセスされるため、外部記憶上の DB に格納されている。

5 万件/秒の処理を行っていること、ならびに、実際の商用システムで稼働していることから、PSA の目指した性能と経済性、ならびに各種性能改善効果の十分性は満たされたと考える。

6.2 QUICK

TAM は、国内最大の金融情報提供会社のシステムである QUICK の次世代情報配信基盤に適用された (図 14)。2011 年 6 月から商用稼働を開始している [4]。

この配信基盤は、東証などから株式に関する大量データをリアルタイムに高速処理する。東証からの情報量増大に

*4 PSM に対するコマンド入力により指示する。

*5 Open Mission Critical System

PSA適用事例：株価配信システム

株式の時価情報をリアルタイムに配信するシステム

大量かつ高速化(低レイテンシー:msオーダー)を実現するため、

PSA(TAM)を使用した処理方式を適用、2011年6月サービスインし稼働中

技術的ブレークスルー：魅力的な価格で、超高性能を提供するため

- ① 新たなPFへの製品対応 ⇒ TAMのLinux対応
- ② 高信頼で安価なオープン製品 ⇒ Express5800/スケラブルHAサーバ、Enterprise Linux with Dependable Supportの採用
- ③ 高速通信 ⇒ 10ギガLAN(マルチキャスト)

・マルチキャスト:ネットワークで、決められた複数のノードに対して、同時にパケット送信する方法。
 cf.ユニキャスト:決められた1つのノードに対して、パケットを送信する方法
 ブロードキャスト:不特定多数に対して、パケットを送信する方法

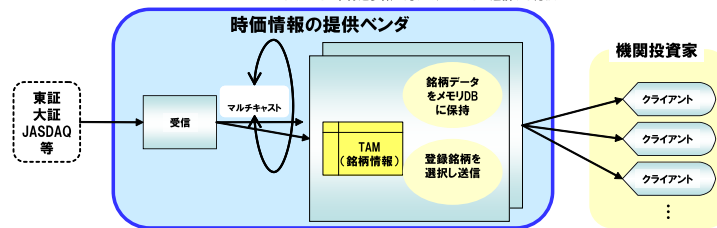


図 14 QUICK 様の株価高速配信の事例

Fig. 14 Stock prices fast delivery system by QUICK.

対応するため、システムアーキテクチャを刷新し、TAMを適用した。大量の情報を受信して、それぞれを数十台のサーバへいっせいに配信する処理をミリ秒オーダーで実現した。

低コストでの構築を目指したので、Linux をベースにした OMCS 技術（高信頼サーバ、Enterprise Linux with Dependable Support）を採用している。

7. 今後の展開：PSA の新たな応用領域

ブロードバンドによる通信能力の大幅な向上とモバイルの進展による多数のクライアントの出現により、ネットワークを介したサービスがあらゆるビジネスで拡大している。Machine to Machine を含むサービス拡大にともない、今後、大量イベント処理への PSA の適用が拡大すると予想しており、以下のような客先提案が行われている。

7.1 位置情報

GPS による位置情報と連携し、新たなサービスを提供することが考えられる。携帯電話や自動車などから GPS を用いた位置情報を大量のイベントとして入力し、利用者の趣味嗜好と提供情報（レストランやドライブインなどの近くにある店舗情報、渋滞情報など）とのマッチングを行い、タイムリに利用者に情報を通知するサービスが一例である。大量イベントの処理には PSM を適用し、位置情報とのマッチングには TAM によって高速化を行うことができると考えられる。

7.2 RFID (Radio Frequency Identification)

RFID は製造業などで部品や製品に付与され、サプライチェーンにより各種企業間で製品が流通する状況をトレ

スするために用いられる。RFID のトレース情報は莫大の量となるので、PSA による大量イベント処理の効果を期待できる。

広義の RFID の一種としては、非接触 IC カードがある。Suica などの乗車カード、Edy などの電子マネーがあり、利用者と購買行動の情報はマーケティングに活用することができると考えられる。

7.3 スマートメータ

欧米や中国だけでなく、日本でも震災後の停電の影響もあり、スマートメータへのニーズが高まっている [5], [6]。

スマートメータから電力消費量をリアルタイムに送信して、個別の電力消費量から使用時間帯による割引計算などの課金処理、各地域の電力消費量を計算し電力供給量の監視や制御、法人に関しては特定な法人に属するスマートメータからの消費電力総和を求め法人への請求明細の作成や法人としての電力使用量の上限監視などを行うシステムが今後のスマートグリッドに求められる。

スマートメータの自動検針システムにおいて、消費電力量は一定周期のタイミングでセンタに送信される [6]。これらを個人、法人ごとに集計し、累計値を管理することで、リアルタイムに地域や法人に関する総電力量を把握することが可能となり、供給電力のコントロールに利用することができる。

以下、スマートメータの自動検針への PSA 適用イメージを詳しく説明する。

スマートメータからは、スマートメータ ID、10 分間の消費電力量などの情報が料金計算センタに送信されることとする。ある電力会社の管理下のスマートメータの数を

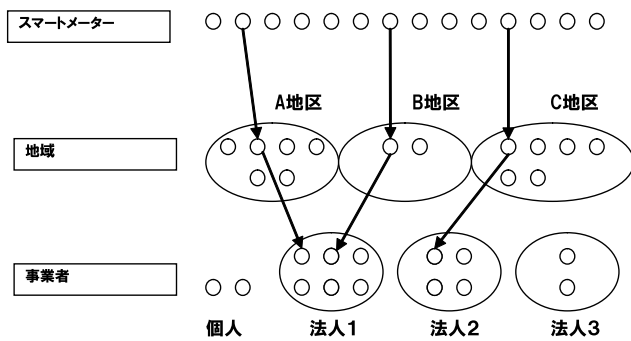


図 15 スマートメータの情報のグループ化

Fig. 15 Grouping of Smart Meter.

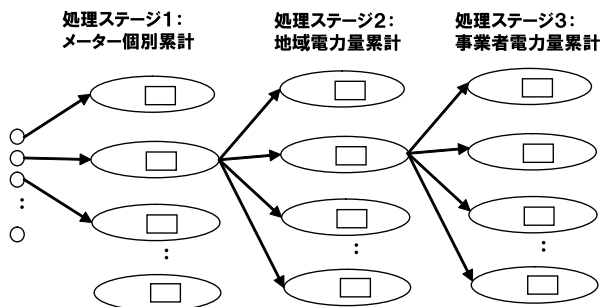


図 16 処理ステージとストリームのイメージ

Fig. 16 Stage and stream.

3,000 万台と仮定すると、約 5 万件/秒のイベント発生量となる。

センタ側の料金処理としては、まず、スマートメータ ID ごとに累計値を計算し、次に、スマートメータが属する地域の消費電力の累計値を計算し、事業者に所属しているスマートメータの累計値を計算することとする。地域の累計値により、どの地域の消費電力が大きくなっているかをリアルタイムに知ることができる。また、事業者ごとの累計値により、法人としての消費電力の傾向を見ることができ、たとえば、リアルタイムに電力消費量の大きな事業者を特定し、節電対策時であれば該当事業者に対して緊急に自粛を促すことができるようになる。

図 15 は、あるスマートメータの情報が、ある地域に属し、またある法人に属している様子を示したものである。

この処理モデルでは、3 階層にステージを分けることができる (図 16)。

スマートメータ単位の消費電力累計処理ステージ、地域ごとの消費電力累計処理ステージ、事業者ごとの消費電力累計処理ステージである。

① スマートメータ単位の消費電力累計

スマートメータから発生するイベントデータを処理するために、並列実行できるストリームを複数定義することとし、PSM はスマートメータ ID により対応するストリームにイベントを渡す。この処理ステージでは、アプリケーションは、TAM のメモリテーブル上のスマートメータごとの累計値にイベントデータ中の消費電力量を加算する。

同時に、累計用とは別の契約管理用メモリテーブルには、スマートメータごとの契約情報が保持されており、アプリケーションは 2 つめの処理ステージで利用する地域 ID と 3 つめの処理ステージで利用する事業者 ID を契約管理用メモリテーブルから取得し、出力イベントにセットする。

地域 ID と地域ストリーム ID との関係を宛先用メモリテーブルとして保持しておき、アプリケーションは宛先用メモリテーブルを検索して、後続のストリームを決定する地域ストリーム ID を取得し、該当地域ストリーム ID を宛先として、PSM にイベント出力を要求する。

② 地域ごとの消費電力累計

この処理ステージでは、1 つの地域ストリーム ID で識別されるストリームは、複数の地域 ID ごとの集計を行う。アプリケーションはイベントデータの中に含まれている地域 ID に従って、TAM のメモリテーブルを検索し、地域 ID に該当する累計値を更新する。またアプリケーションは、イベントデータ中に含まれている事業者 ID と事業者ストリーム ID を特定するための宛先用メモリテーブルを用いて、後続のストリームを決定する事業者ストリーム ID を取得し、事業者ストリーム ID を宛先としてイベントの出力を PSM に要求する。

この処理ステージのストリーム分割は、属するスマートメータの数が一定になるように分割するのが分かりやすいだろう。

③ 事業者ごとの消費電力累計

この処理ステージでは、1 つの事業者ストリーム ID で識別されるストリームは、複数の事業者 ID ごとの累計処理を行う。アプリケーションは、イベントデータの中に含まれる事業者 ID を用いて TAM のメモリテーブルを検索し、事業者 ID に該当する累計値を更新する。

この処理ステージでのストリーム分割は、注意が必要である。事業者ごとに所属するスマートメータの数にばらつきがあるので、あるストリームに処理が集中しすぎないように、各ストリームへのイベント流入量を見込んで分割しておくのが望ましい。

このようにして、3 つの処理ステージを経て、3 つの累計値の集計が完了する。

7.4 業務要件に関する再整理

PSA の新たな活用領域を考えるにあたり、本論文で採用した比較的単純な業務モデル以外のモデルについて、考えてみる。

本論文の中では、イベント発生源をいくつかの種類にグループ化して集計するケースを想定した業務例をもとに説明したが、イベント発生源ごとにグループ化することは必須ではなく、「業務要件に従って、なんらかの複数の観点によるグルーピングが行われる処理を、複数の処理ステージに分解することで、資源の待ちをなくし並行動作可能に

できる業務」であれば、PSA を適用し、高い処理性能を期待できることは明らかであろう。今後、様々な業務が PSA のアーキテクチャで実行されることと考える。

8. まとめ

PSA はスーパー OLTP の開発が着想ポイントであるが、近年のメモリ処理は DWH のリアルタイム処理など大量の DB の並列処理でのメモリ処理の研究が行われている。

PSA がメモリ処理にもたらした意義を整理すると、以下のとおりである。

① 大規模メモリ処理の実現

当時（2003 年）のサーバは、512 GB のメモリを搭載可能であり、この大容量メモリを使用して実用に耐えうるメモリ DB (TAM) を実現した。

② 新しい OLTP（スーパー OLTP）

スーパー OLTP として扱う大量イベントをリアルタイムに処理するための新たなアーキテクチャの幕開けとなった。

③ 世界最大級の実績

世界最大規模の料金システム（ドコモ）、大規模な株価配信システム（QUICK）の実績により、ミッションクリティカル性が必要とされる大規模システム領域におけるスーパー OLTP として PSA の実効性を確認することができた [3], [4]。

以上、それまでの OLTP モニタ、DB システムによる OLTP に比べ 10 倍以上の高性能なスーパー OLTP を、メモリ処理を駆使して実現した PSA のアーキテクチャの具体的な構成と特徴を述べた。

スーパー OLTP が対象とする業務処理の領域は、図 17 に示すとおりとなる。インターネットのユビキタスコンピューティング時代にはこういったタイプの OLTP の増加が予想され、その先駆けとなるシステム事例である。

今後ユビキタスコンピューティングの時代には、さらに重要性が増すと考え、今後の新たな応用例（GPS、RFID、スマートメータ）にも触れた。OMCS の適用範囲も既存の OLTP の世界から PSA を活用した新しい OLTP（スーパー OLTP）への発展が期待され、PSA の重要性は増すも

のと考えている。

謝辞 OMCS のシステム開発において PSA システムモデルの必要性をご教示いただいた NTT ドコモの皆様と PSA 上のアプリケーション開発に取り組まれた NTT データの皆様の絶大なるご支援がなければ PSA システムモデルの開発は不可能であった。深く感謝申し上げる。また、PSA の具体的開発とそれを適用したシステムの開発に参画された皆様にも重ねて感謝申し上げます。

参考文献

- [1] 相澤正俊, 東 健二, 川浦立志: OMCS 構築技術の研究, CDS 研究会, CDS3-11 (2012).
- [2] 相澤正俊, 東 健二: OMCS のシステムモデルによるプラットフォーム構築, CDS 研究会, CDS3-12 (2012).
- [3] 穴戸周夫: 経営の見える化を追求—NTT ドコモが取り組むリアルタイムマネジメントシステムの実態, ZD Net Japan, 入手先 (<http://japan.zdnet.com/cio/sp-09oow/20392427/>) (参照 2009-04-27).
- [4] ソフトバンクビジネス + IT: 金融情報ベンダー QUICK, 次世代情報配信基盤に NEC を採用, 入手先 (<http://www.sbbt.jp/article/cont1/23723>) (参照 2011-08-19).
- [5] 日本経済新聞: スマートメーター, 年 2 億台の大市場へ, 入手先 (<http://www.nikkei.com/tech/ecology/article/>) (参照 2011-06-06).
- [6] 宮内直人, 水野忠則, 峰野博史: スマートグリッド監視・管理システム, 情報処理学会研究報告 MBL, 2011-MBL-59(10), pp.1-7 (2011).
- [7] Arasu, A. et al.: STREAM: The Stanford Stream Data Manager, *IEEE Data Engineering Bulletin*, Vol.26, No.1.
- [8] 森 有一, 角谷有司, 西澤格ほか: 情報爆発時代の到来に向けた大量高速データ処理技術への取り組み, 日立評論, Vol.90, No.7, pp.612-613 (2008).
- [9] 松浦紘也, 鈴木豊太郎: データストリーム処理とバッチ処理における動的負荷分散, 電子情報通信学会技術研究報告 DE, データ工学, Vol.110, No.107, pp.69-74 (2010).
- [10] 富山卓二: 次世代ネットワーク時代におけるバックエンドシステムのアーキテクチャ, NEC 技報, Vol.59, No.2, pp.45-48 (2009).
- [11] 窪田 晃: リアルタイム処理システム, 処理装置, リアルタイム処理方法, 及びプログラム, 特許第 4232109 号 (2008.12.19).
- [12] 宮田 剛: トランザクション処理装置, トランザクション処理方法, 特許第 4277873 号 (2009.3.19).

商標について

OMCS, BankingWeb, PSA, ECOCENTER は, NEC の日本国内における登録商標です。OpenDIOSA, PARALLELSTREAMMONITOR, SIGMABLADE, InfoFrame, WebOTX は, NEC の日本国内およびその他の国における登録商標です。

Windows, SQL Server は, 米国 Microsoft Corporation の米国およびその他の国における登録商標です。UNIX は, The Open Group の登録商標です。HP-UX は, 米国およびその他諸国における Hewlett-Packard Company の登録商標です。Oracle, Java, WebLogic, TUXEDO は Oracle

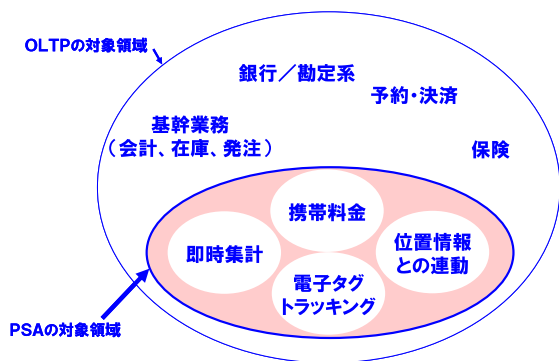


図 17 PSA (スーパー OLTP) の適用領域

Fig. 17 Target area of PSA (super OLTP).

Corporation およびその関連企業の登録商標です。SAP はドイツおよびその他の国における SAP AG の商標または登録商標です。Linux は、Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。その他の名称は、それぞれの所有者の商標または登録商標です。



相澤 正俊 (正会員)

1971 年東北大学大学院工学研究科電気通信工学専攻修士課程を修了し、1972 年日本電気株式会社入社。主に基本ソフト (OS) 開発と IT ソリューション事業を担当。2008 年代表取締役執行役員副社長に就任、2011 年より株式会社国際社会経済研究所 (NEC グループ会社) 理事長に就任。



富山 卓二 (正会員)

1975 年静岡大学理学部を卒業し、日本電気株式会社入社。主に基本ソフト (OS) 開発と IT ソリューション事業を担当。2010 年より取締役執行役員常務に就任、2011 年より NEC システムテクノロジー株式会社代表取締役執行役員社長に就任。



斎川 幸貴 (正会員)

1985 年法政大学工学部電気工学科卒業。同年日本電気株式会社に入社。以来、メインフレームの OS 開発、UNIX の開発を経て、数々の OMCS 関連の大規模システム構築に従事。