ホワイト・ペーパー

ビジネス・アクティビティ・モニタリング（BAM）：エンタープライズにおけるプロセス制御

Tom Lubinski, SL Corporation

February 12, 2008

ビジネス・アクティビティ・モニタリング（BAM）アプリケーションは一般的になってきており、また複雑化しています。リアルタイム・データの解析と可視化についての多くは、プロセス制御やテレメトリ・システム前世代との比較から学ぶことができ、似ている点も多々ありますが、リアルタイム・データの操作に伴う技術には顕著な違いもあります。この白書は、BAM アプリケーションにおける解析と可視化における実践上の課題について解説するものです。そして、プロセス制御の業界でリアルタイムな可視化と監視技術を 25 年間提供してきた経験から得た教訓が、いかに今日のエンタープライズにおけるより効果的で成功するBAMソリューションの提供に適用できるかを解説していきます。

I. はじめに

リアルタイム・データの解析と可視化は、新しいことではありません。生産自動化やプロセス制御システム、ネットワーク管理アプリケーションやテレメトリ・システムでは、昔から使用されてきたコア技術です。

最近では、高度なビジネス・アプリケーションにおける監視と自動制御への関心が高まっています。サプライチェーンや顧客関係管理（CRM）、ビジネス・インテリジェンス（BI）など、分散するアプリケーションの統合が、新たな課題をつくったからです。

そのようなエンタープライズ・ソフトウェア・アプリケーションは、多くの場合自動化されており、色々な面で工場や生産設備に例えて見ることができます。たとえば、バックグラウンドで稼動する一連のアプリケーションによって実行される、Web サイトにログインした顧客からの受注・注文処理・請求では、マニュアルな介入はほとんど、またはまったくありません。

このようなオペレーションの活動・健全性・状況を監視することの呼び名の一つとして、ビジネス・アクティビティ・モニタリング（BAM）が知られるようになりました。実際のところ、これらのシステムにおける目的は、イベント駆動型の状況判断を提供することであり、どの時点においても、オペレーションとそれをサポートしているシステムの現況に可視性があります。

これらのシステムにおけるアクティビティのリアルタイムな監視は、製造業における設備の監視と制御にとても似通っています。両者において、リアルタイムなデータ収集、データの解析、データのさまざまな形式によるプレゼンテーションが必要だからです。しかしながら、これらの機能の一部は、両者間で大きく異なります。

これらの違いから、従来のプロセス監視と制御で使用されていたレガシー・システムを、新たに出て来たエンタープライズ監視アプリケーションにうまく適用できないため、新興の製品と技術がその新しい要求条件を満たすために投入されているわけです。

本白書は、ビジネス・アクティビティ・モニタリング（BAM）システムにおけるリアルタイム・データの操作におけるさまざまなアプローチについて紹介し、ユニークと言えるリアルタイム・データの性質と必要とされる解析と可視化の種類についての理解を支援することを目的としています。そしてリアルタイムで多次元のデータ処理における重要な要求条件と技法について、特に解説するものです。

本白書の著者と SL 社は、アプリケーションの監視と可視化で 25 年の経験を持っており、また Java のエキスパートでもあります。同社の RTView 製品は、多種多様のアプリケーションによって生成されるリアルタイム・データの操作に特化しており、ビジネス監視アプリケーションに共通するほとんどの要求条件を満たす機能を備えています。

従来のプロセス制御分野と、最新のエンタープライズ・アプリケーションの両方における経験が、これらシステムにおける要求条件に対し、ユニークな見解と解決策を提供しています。

II. ビジネス・アクティビティ・モニタリング(BAM)

ガートナー社による著名な定義では、ビジネス・アクティビティ・モニタリング（BAM）システムは、クリティカルなビジネス・パフォーマンス指標へのリアルタイムなアクセスを提供するもので、ビジネス・オペレーションのスピードと有効性を改善します［1］。 BAM アプリケーションは、リアルタイムに複数のデータソースからイベントを受信して評価指標を更新し、状況を評価して結果をダッシュボード形式で出力し、複雑なイベントまたはアラートを発信します［2］。

また、従来のビジネス・インテリジェンス（BI）と BAM の間では、主に「リアルタイム」の定義の違いについての議論も若干出ています［3］。 さらに、BAM と多少重複する他アプリケーション・タイプも多くあります。

純粋な BAM ソリューションは、多数データソースの上流エンド、特に下流から得た主要業績評価指標（KPI）に、焦点を置いていると言えます。しかしながら、多くの場合、ビジネス・オペレーションのスピードと有効性を改善するには、これをサポートするデータのナビゲーションと解析を組み合わせることが、必要です。

この理由から、また本白書の目的から、BAM は、ビジネス・サービス管理（BSM）やインフラ監視（I M）と呼ばれることがある下流レベルの監視の多くを、包括または包含していると言えます。次のテーブルは、それぞれのレベルにおける特徴をまとめたものです：

表1. 監視レベル

レベル	ソース	KPI
ビジネス・アクティビティ・モニタリング（BAM）	カスタムまたはパッケージ・アプリケーション	Webサイト・ビジター数、予約されたチケット数、売上成約率
ビジネス・サービス管理(BSM)	メッセージング・ミドルウェア、BAMエンジン、Webサービス、カスタム	送信メッセージ数／秒、プロセス間におけるメッセージ遅延
インフラ監視(IF)	コンポーネント、プロセッサ、ルータなど	CPU使用率、使用可能なメモリ容量、送信ネットワーク・パケット数

総じて、最も純粋な意味での BAM におけるリアルタイム・データとは、Web サイト・ビジター数や予約されたチケット数など、業務をサポートしているアプリケーション・プロセスによってビジネス・データに変換された内容を表すもの、と言うことができます。そして、送信されたメッセージ数などの下流レベル・データは、事業をサポートしているサービスまたはインフラストラクチャの健全性と状況に関する情報を表します。

しかしながら、多くのケースでは、実際にはこれらはすべて同じで一つです。たとえば、オンライン Web ショップでは、サポートしているプロセスの遅延は、そのサイトで販売された商品の数に直接関係しています。もしユーザがシステム遅延を感じると、他のサイトに移ってしまうからです。この場合、BAM アプリケーションは、すべてのレベルからのデータを解析して提示しなければなりません。このことから、BAM がビジネスをサポートしているサービスの監視（BSM）を包括している、と言えます。

それぞれのレベルでは、関係するアプリケーションはリアルタイム・データを生成します。これらのデータは、アプリケーションの監視に使用でき、システムがどのように稼動しているかの状況を把握し、パフォーマンスを最適化することが可能になります。これが、BAM の明確な目標です。

III. リアルタイム・データソース

過去数十年間、リアルタイム・データの監視は、主にテレメトリやプロセス制御といった特殊なアプリケーションに関係があるものでした。これらのアプリケーションでは、温度センサーやレベル・インディケーターなど個々のコンポーネントが、定期的な間隔で測定値を生成し、OPC（OLE for Process Control）などの業界標準プロトコルを使って、中央のサーバに伝送されます［5］。

数千以上のデータ・ポイントを解析してアーカイブし、トレンド解析、アラーム生成、異常事態の検知を可能にすることによって問題の発生を未然に防ぐ、高度な技術が開発されました。さらに、リアルタイムな測定値をベースに、プロセスをダイナミックに調整する品質管理機構も装備しました。

これに比較すると、今日のビジネス・アプリケーションの監視は、原始的と言えます。一つの組織における顧客関係や受注処理といったさまざまな機能を支援するアプリケーションの数々が独立して開発されています。そして、リレーショナル・データベースが、エンタープライズ・アプリケーション開発の基盤として使われていますが、スプレッドシートのような数々のデスクトップ・アプリケーションが、事態を複雑化しています。独立したアプリケーションの孤島が、重複データで溢れているからです。

ゆえに、複数のシステムを結び付け、ビジネス・プロセス・アプリケーションの対話処理を自動化することは、とても大きなビジネスになりました。数多くの企業が、企業内アプリケーション統合（EAI）やビジネス・プロセス管理（BPM）製品を、サポート・サービスとともに提供しています。しかし、これらの製品はビジネス・アプリケーションの統制を図るのが一般的であり、リアルタイムな監視機能については、大して提供していません。

そして、これら多くのシステムの核となっているのが、メッセージ指向ミドルウェア（MOM）です［6］。BEA Systems 社、TIBCO Software 社、IBM 社などは、構造化されたデータ・パケットをネットワークの周りに伝送し、アプリケーション間で共有できるメッセージング・システムを開発しています。Java メッセージ・サービス（JMS）は、これらシステムのほとんどがベースとして構築している新しい業界標準であり、アプリケーション間におけるリアルタイムなデータ・ストリームを、初めてサポートしたものです。

プログラム・トレーディングなどの金融サービス・アプリケーションでは、パフォーマンスが最も重要です。初期の MOM システムの性能を桁違いに改善する新しい技術が登場し、ビルトインされた解析と複雑なルール処理を提供するようになりました。これらの複合イベント処理（CEP）［7］システムは、自動化された不正検知といった、より高度なアプリケーションを可能にしています。

その他のリアルタイム・データソースとしては、アプリケーション内に管理測定機能を持たせる新しい業界標準となった Java 管理拡張（JMX）［8］を実装したアプリケーションがあります。定期的にポーリングを実行すれば、リレーショナル・データベースもリアルタイムに近いデータを提供することができます。さらに、Big Brother などのシステムから、リアルタイムなインフラ監視データが使用可能です。

これら数多くのリアルタイム・データソースにアクセスし、解析して内容をプレゼンテーションすれば、ビジネス・アクティビティ・モニタリングのシステムを提供することが可能です。それぞれは膨大な生データ源であり、それぞれのコンテンツにはビジネス・オペレーションの健全性と状況に関する豊富な情報が含まれています。

IV. 多次元のデータ構造

さまざまなタイプの監視システムから得られるリアルタイム・データの性質は、長い年月とともに変化してきました。従来のプロセス制御とテレメトリ・システムでは、それぞれが個別のタイムスタンプを持った多数のデータ「ポイント」によって運用されます。それぞれがタイムスタンプを含んでいるという点に注意してください。

表2. 個別のデータ・ポイントの例

時間	バルブ1
7:01	開
7:02	開
7:03	閉

時間	バルブ2
7:01	開
7:02	閉
7:03	開

一方、メッセージング・ミドルウェア・システムでは、関係したデータ・ポイントを一つのメッセージとしてグループ化して伝送するというニーズによって進化してきたものです。たとえば、アプリケーション間の一つのメッセージには、一つの注文に関する情報が含まれており、その購買者、商品の数、価格、他の関連したデータを包含しています。典型的な一つのメッセージには、複数の個別のデータ・フィールドで構成される一つの「行」から成り立っており、その行全体に共通の一つのタイムスタンプを持っています。

表3. 構造化されたデータの行

タイムスタンプ	支社	エージェント	予約	キャンセル	失敗
7:05	北	メアリー	1	0	0
7:05	東	ジョー	2	1	1
7:10	東	ボブ	0	1	1
7:15	北	メアリー	2	2	0
7:15	東	ボブ	1	1	0

エンタープライズ・アプリケーションの開発者にとっては、一つのメッセージが複数のフィールドで構成されていることは明白で、単に構造化された一つのデータ・セットです。しかしながら、これはプロセス監視制御アプリケーションの一般的なデータとは、かなり異なります。その結果、リアルタイムなプロセス・データの収録、伝送、解析、アーカイブに使われる従来のリアルタイム・プロセス・データは、エンタープライズ・アプリケーションの監視には適用しにくいわけです。これら2つのタイプのシステムをさらに深く調査していくと、多くのことを学ぶことができますが、データ処理においては新しい技術の開発が必要であることがわかります。

ビジネス・アプリケーションのデータでもう一つ重要な側面は、データ・フィールドの性質です。上の例では、次の表4のように、フィールドを時間、インデックス、データの列にまとめることができます:

表4. カテゴリ別のフィールド例

時間	インデックス		データ		
タイムスタンプ	支社	エージェント	予約	キャンセル	失敗

すべてのリアルタイム・データは、メッセージ・ストリームまたはポイント・データから来ているか否かを問わず、必ずタイムスタンプを含みます。メッセージはある特定の時間に着信し、またはそのメッセージにタイムスタンプが含まれていて測定がいつ実行されたかを示します。これは、トレンド解析の実行に必要です。

「インデックス」フィールドは、多くの場合、データソースに関連しています。たとえば、予約システムにおいては、これらのフィールドは支社とその支社内のエージェント(代行者)が含まれます。また、違うアプリケーションでは、ユーザが注文を入力したウェブサイトのページです。そして概ね、これらのフィールドはどのデータが解析されるかによるものです。たとえば、ビジネス・ウェブサイトのアクティビティの監視では、サイトのページ毎に、またはそのサイトに入ってきたIPアドレス毎に、それぞれどれくらいの注文があったかを知ることです。

「データ」フィールドには、実際の測定値が含まれます。これらは、メッセージに格納された値です。多くの場合、同じ時間間隔で得られたサンプルには、一つ以上の値があります。たとえば、最終間際の1分間に実行されてうまく入った予約の数と、うまく入らずに失敗した予約の数は、同じメッセージ内になります。これらは関連性があるものの、指標としては異なるものです。

以上のような形で説明できるデータを、多次元と呼びます。単一次元のデータと異なり、一つの共通のタイムスタンプで関連したデータが、いくつかの列で構成されるということです。そして、ビジネスの理解には、これらの異なる多次元におけるデータの「スライス&ダイス」が必要になってきます。

そしてこの多次元の性質が、データをどのように可視化して解析するかに大きく影響します。

V. 可視化

従来のプロセス制御における可視化アプリケーションと BAM における可視化アプリケーションには、参考になる類似点があります。両者における共通の要求条件の一つは、ビジネスそのものまたはそれをサポートするサブシステムであるか否かを問わず、監視対象システムの現時点での状態を示すことです。

A. BAM 可視化の性質概要

従来のプロセス制御システムにおける可視化では、多くの場合、模擬ダイアグラムが含まれます。つまり、実際のプロセスまたは監視しているプラントを模擬する画面です。これらの画面では、下図1のように、温度や圧力などをはじめとする測定値とともに、プラント・システムのバルブや制御装置それぞれの現在における状態を表示します:

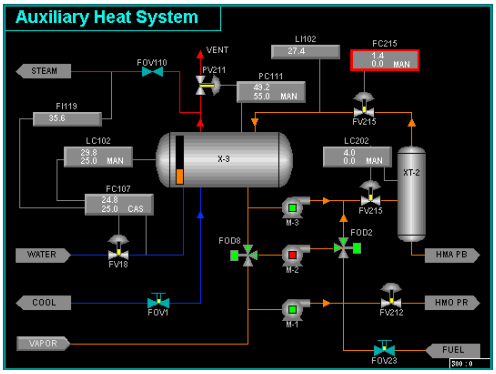

図1. 典型的なプロセス制御における模擬ダイアグラムの作成

一方、ビジネス・アクティビティ・モニタリング(BAM)においては監視されるプロセスが通常、抽象的です。模擬ダイアグラムで実際に表現できるボイラーやパンプ、センサーといった物はありません。

ビジネス・プロセスは、ソフトウェア・アプリケーションの一つのサブシステムからもう一つのサブシステムにデータが漏斗のように注がれることです。そのようなプロセスは「バブル(泡)」や円で表現されることが多く、その中にプロセスの性質を示唆するアイコンまたはイメージがあります。これらのプロセスの多くは、下図2のように接続して表現します:

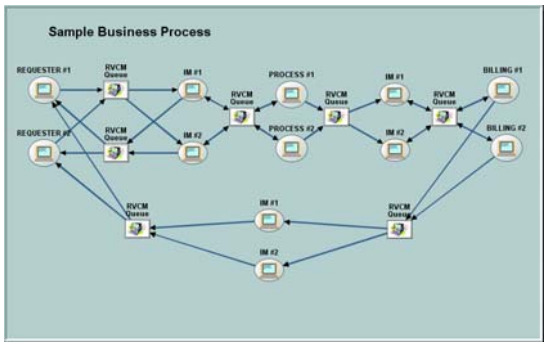

図2. ビジネス・ダイアグラムの例

プロセスの状態は、値、色、バブル内のアイコンによって、表現されます。たとえば、プロセス・ステップ間における遅延状態であったり、処理されたオーダーやリクエストの総数であったりします。センサー値の代わりに、プロセス・ステップ間における時間差を表示する計算された値のセットになります。そして、この情報はバブルを接続している線上で表示することができます。

このように見ていくと、ビジネス・プロセス画面のバブル・ダイアグラムは、従来のプロセス監視システムの模擬ダイアグラムとは異なることがわかります。違いは、表示されている情報が物理的なハードウェア要素の状態ではなく、抽象的なソフトウェア・プロセスの状態であるということです。

しかしながら、これら2つのタイプのシステム間には、多くの類似点もあり、大変参考になります。一つは、画面のパラメータ化です。これは、可視化における最もチャレンジングな課題であり、また見過ごされやすい課題でもあります。そして、最も成功しているプロセス制御システム製品は、このパラメータ化において優れた技法を提供しています。

B. 画面のパラメータ化

リアルタイム・データの可視化で使う画面では、データ要素の値が画面のオブジェクトに表示されるように参照して設計しなくてはいけません。異なるデータソースで、一つの画面を複数回使用するためには、それらの参照は間接的である必要があり、パラメータ化しなくてはいけません。

パラメータ化には、二つの種類があります。一つは、特定のプロセスに関係したいくつかの値を見せるために設計する合成グラフィカル・オブジェクトです。このオブジェクトは、同じダイアグラムに何回もインスタンスでき、それぞれには異なる一行のデータが接続され、テーブル内にあるインデックス列の一つ(たとえば、一つの支社)によってユニークに識別されるものです。そのような画面の指定では、正しいデータを選択するために使用されるインデックス値が、オブジェクトの描画時に渡されます。

この例を下図3で示します。ここでは、それぞれのサブシステムについての情報は、テーブルの一行でデータを表すアイコンにまとめられています:

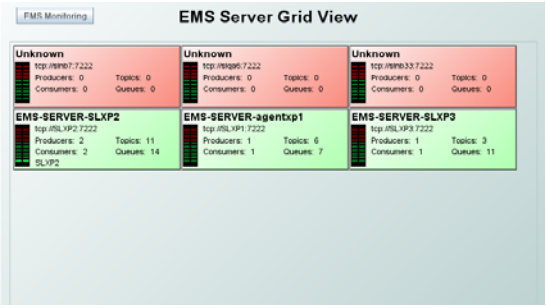

図3. 複数のデータ行に接続されるアイコンを表示するオブジェクト・グリッドの例

二つ目のケースでは、高レベルな画面に示される問題の原因を判定するために、より詳細なデータにドリルダウンする必要があります。

ユーザが画面のオブジェクト一つを、またはデータ・テーブルの一行を選択したときに、選択されたオブジェクトを駆動するパラメータの値をシステムがピックアップできるよう簡単に構成でき、プロセスのドリルダウンで起動される新しい画面にパラメータとして渡せる方法が必要です。そのパラメータは、選択されたオブジェクトに特定された詳細のデータを表示するのに使います。

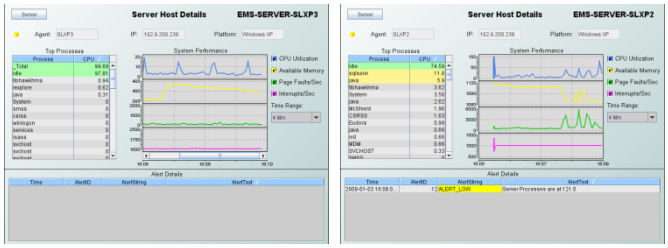

図4. 異なるデータソースからのデータを表示する同じ画面

効果的な BAM システムにおいては、上記両方のシナリオが重要です。パラメータ化のサポートがなければ、一つの画面から、それぞれが特定のデータを参照する複数のコピー画面を構築しなくてはならないからです。ドリルダウンのケースでは、パラメータ化によるナビゲーションがサポートされていない仕組みでは、選択とパラメータ化を行なう工程に、かなりのプログラミングを要することになります。

データ変数をリアルタイムに示すトレンドグラフをポップアップするのは簡単です。しかしながら、さまざまな用途で再利用できるようにパラメータ化した画面の開発は多大な工数を要し、この点が大いにして見落とされがちです。

VI. リアルタイム・データの永続化

リアルタイム・データは一時的なものです。つまり、入って来たリアルタイム・データは、次の記録が入って来ると、消えてしまいます。次々と入って来るリアルタイム・データの有効な解析とプレゼンテーションには、それぞれのデータ行を操作または以前のデータと比較するために格納しておく必要があります。これには、一時的なリアルタイム・データを格納またはアーカイブし、「永続化」しなくてはなりません。

リアルタイム・データの永続化は、データベース・アーカイブと混乱されることが多くあります。完全な永続化の実装にはデータベース・アーカイブ機能は不可欠ですが、多次元データにおいては、さらに次に述べる重要な機能が要ります。

A. 現在値テーブル

BAMシステムで見られる概ねのリアルタイム・データは、非同期に入って来ます。イベントが起きると、メッセージが送信され、すぐに受信されて処理されます。異なるデータソースから入って来るイベントは、異なる時間に、それぞれが「同調することなく」発生し得ます。

リアルタイムな多次元データの複数行が非同期に入って来ると、問題が一つあります。たとえば、新しい注文を登録しようとするイベントを複数の支社オフィスが送信すると、すべての支社からのデータを一度に見る簡単な方法がありません。それぞれのデータ・イベントは、一支社からの情報のみを含んでおり、次のイベントが入って来ると、オーバーライドされてしまうからです。

この問題を解決するためには、それぞれのデータソースをユニークに識別化できるよう、データの行に列でインデックス化した「現在値テーブル」を、システムで管理する必要があります。インデックスは、支社といったような一つの列で構成されている場合もありますが、支社とエージェントといった複数列で組み合わさっている場合もあります。それぞれのデータ・イベントが入って来ると、現在値テーブルにおける特定のスロットに格納されます。これによって、それぞれユニークなインデックスで記録された最新のデータへのアクセスが提供されます。

表5. 現在値テーブルの例

支社	エージェント	予約	キャンセル	失敗
北	メアリー	4	2	0
北	サム	4	1	0
東	ジョー	0	2	1
東	ボブ	1	1	2
東	メアリー	2	3	1

C. 項で述べるように、リアルタイム・データをリレーショナル・データベースにアーカイブするのが一般的です。しかしながら、リレーショナル・データベースを使って、現在値テーブルを管理するのは難しいものがあります。個々のデータソースの最新値を判別するのに、大量の履歴データに対して、常にクエリを実行し続けなければならないからです。

これに代わってインメモリで現在値テーブルを構築できることが、完全なBAMソリューションには欠かせない重要な機能となります。

そして、現在値テーブルを複雑化する数多くの要求条件があります。まず、テーブル内でアクティブではなくなったオブジェクトは、次々と削除されないとテーブルが大きくなり過ぎてしまいます。また中には、完全に独立したサブシステムの閲覧画面に切り替えるときにテーブルを一旦クリアにしなければならないアプリケーションも、あります。

LABView [9] などのプロセス制御アプリケーションでは、主に現在値への分散したアクセスを実現するために、現在値テーブルを提供しています。BAMアプリケーションの実装では、この基本的なコンセプトに、多次元データの処理を加えて拡張したものになります。

B. インメモリ・キャッシュ

現在値テーブルは、それぞれのインデックス化されたデータソースから、最後のデータ行へのアクセスを効率的に提供します。さらに、最後の行だけではなく、それより前の行もある範囲内で保持することができれば、それは「インメモリ・キャッシュ」として参照することができます。

現在値テーブルのように、最後の行より前のインメモリ・キャッシュ・データもインデックス化されていなければなりません。キャッシュの長さは、行数またはストレージが使用可能な時間範囲を指定することによって、コントロールします。

リアルタイム・データは、その性質上、時間で連なったデータで、時間によって順序だった一連の測定値を意味します。そのようなデータの解析は、時系列でデータをプロットすることによって行われます。

良くできたインメモリ・キャッシュ・システムは、格納された時系列データを限られた時間枠で効率的にアクセスすることができます。これは、リアルタイムにデータを解析するのには不可欠です。たとえば、問題の可能性があることをオペレータに警告するためには、短期における異常な変化の傾向を検知しなければならないからです。

C. リレーショナル・データベース・アーカイブ

ほとんどのBAMデータソースから入って来るリアルタイム・データは行形式であるため、「ヒストリアン」プロセスを使って、リレーショナル・データベースにアーカイブするのが自然に見えます。この利点としては、履歴データを処理して記録を生成する、共通のレポーティング・ツールとして使えることです。

また、アーカイブのもう一つの目的は、データを見ていたプログラムが何らかの理由で終了して復帰した場合に、終了した時点でのデータの状態をリトリーブできることです。

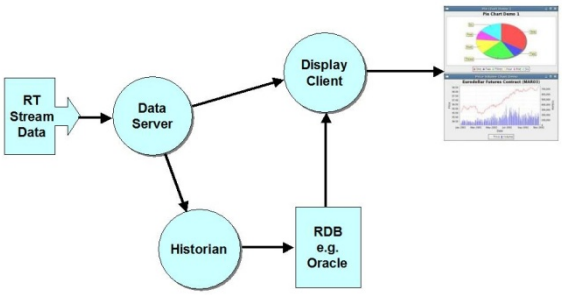

図5. リレーショナル・データベースにアーカイブされるリアルタイム・ダッシュボード

上記のアーキテクチャでは、ディスプレイ・クライアントにおける真のリアルタイム・パフォーマンスを得ることを可能にします。リアルタイム・データへのアクセスすべてをバッファリングするために、データ・サーバ・プロセスが使用されています。ヒストリアンとディスプレイ・クライアントの両方が、データ・サーバに直接アクセスすることができ、表示またはアーカイブしたいデータを取得できます。ディスプレイ・クライアントは、データベースでSQLクエリを実行することによって、アーカイブされた履歴データを得ることができます。

このアプローチにおける一つの問題は、複数のリレーショナル・データベースが SQL 言語定義の一部しか共有しないことです。すべてのデータベースにポータブルな方法で、履歴データに対して時間範囲のデータ集約を実行することはできません。これが、インメモリ・キャッシュを使ってそのような機能を実行するもう一つの理由です。

この応用例の一つが、時系列関数をインメモリで実行する OLAP データベースの使用で、時系列処理におけるより高度な関数が提供されています。ただ一つ、OLAP クエリの文法とキューブの構成は、ほとんどのユーザにとっては複雑過ぎるということです。実際、SQL 開発者に比べると、OLAP 開発者は非常に少ないのが現状です。

D. 特別目的のヒストリアン・データベース

従来のリアルタイムなプロセス制御アプリケーションでは、アーカイブは進化して確立された技術で、数々の会社が時系列データのアーカイブを専門とし、特別な目的を持ったヒストリアン製品を出しています。OSIsoft PI System [10] も、数々のプロセス制御アプリケーションで使用されている最も一般的なヒストリアン・データベースの一つで、AspenTech 製品も同様の製品を提供しています [11]。

これらのシステムにおける共通の機能は、データ・ポイントの値が変わっていない多数のデータ・ポイントを一つの記録に圧縮できる機能です。また、大量のデータの格納に必要なメモリ量を削減することを主な目的に、データを平均化する機能があります。

BAM アプリケーションで見られるリアルタイム・データは、典型的な多次元データです。それに対して、従来のプロセス制御におけるヒストリアンは、単一のデータ・ポイントを扱います。これが、使用上で複雑さを増します。複数の関連した測定値は一つの記録としてまとめなくてはいけませんが、ポイント・データベースではこれを容易に実現することができません。その結果、これらの製品は BAM 市場におけるアーカイブでは使用されていません。

BAM 市場でやや見られる製品の一つは、ラウンド・ロビン・データベースまたは RRDtool [12] と呼ばれるオープンソース製品です。RRD 製品も、時系列でデータを圧縮します。そのシンプルさが貴重な選択肢になる場合もあります。しかしながら、これもポイント・ベースのシステムです。

OSI PI System など、従来のプロセス制御におけるヒストリアンは、ミッション・クリティカルなアプリケーショの数々で使用されてきました。フル機能を完備した BAM システムは、確立されたこれらのシステムで不可欠とされる重要な機能を含まなくてはいけません。そして、さらに複雑な多次元データをサポートする必要があるのです。

VII. 解析

入って来るリアルタイム・データの解析処理は、可視化プロセスの一部であることが多々あり、リアルタイム・データの解析と可視化は極めて関連しています。

解析とは、たいてい、平均、合計、最小二乗法トレンドなどが想定されます。これらは基本的な解析関数であり、良く使われるものです。これらを BAM ソリューションに組み込むことは容易ですが、これらとはまったく異なるタイプの関数を必要とします。それは、数学的なデータ・トレンド解析結果よりも遥かに役立つ情報を、簡単な集約や内訳（統計）表示関数で抽出できるものです。

複合イベント処理(CEP)エンジンの機能によって、最適な BAM が構築できるという提案も出ています。確かに、オンライン不正行為の検知など、極めて複雑な時間との相関関係と判定を要する複雑な CEP アプリケーションにおいては、適合します。

実際には、BAM における解析と可視化において最も望ましいソリューションのポイントは、解析関数と結果の表示に使用されるグラフィカル・オブジェクトの密接な連結であることが実証されています。関数は、多次元データの集約ならびに内訳（統計）表示を提供し、グラフィカル・オブジェクトの属性はその解析結果を直接表示します。

解析関数は、基本的には表データを一つの形式からもう一つの形式に変換するものです。そして、BAM アプリケーションにおける大半の要求条件は、データの集約と関数をさまざまな形式で提供することで、満たすことが可能です。

たとえば、入って来るレンタカー予約のデータ・ストリームは、インメモリ・キャッシュに格納できます。すべてのデータは時系列のテーブル構成で使用可能です。ビルトインされた時間範囲の集約関数を使って、そのテーブル・データの数を、日別と予約エージェント別の内訳で表示します。その結果テーブルのデータは直接棒グラフに供給され、ユーザにとって重要な情報がダイナミックに表示されます。

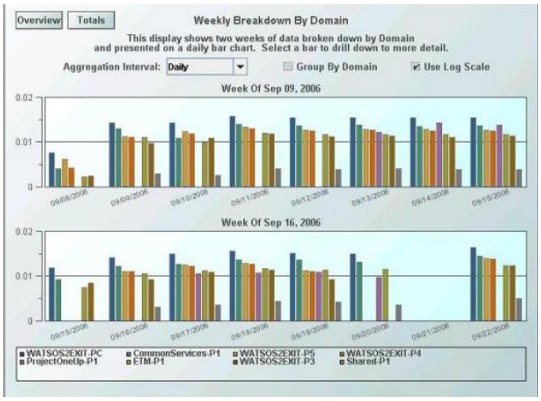

図6. 棒グラフで内訳を2つの次元で表示

このアプローチの利点は、複雑な BAM アプリケーションの開発におけるプログラミングを大幅に削減または削除できることです。解析ならびに可視化パターンのコレクションが提供されており、遭遇するほとんどのケースが、これらパターンの変形に過ぎません。そして、単に異なるデータソースを指定するだけです。

次に、共通してみられるこれらのパターンのいくつかを示します。他にも多数ありますが、これらが最も代表的と言えます。

A. データの集約と内訳表示（統計）— Group By 集計

SQL の Group By 集計のように、これらのパターンは、複数のインデックス列を含むデータ・テーブルに対して操作し、データをさまざまな方法で集約するものです。ユーザは、特定のインデックス列一つまたは、複数のインデックス列セットに対して、合計(sum)、平均(average)、最大値(max)、最小値(min)などを実行できます。

これらのパターンは、インデックス列における次元の数と、データを表示するのに使用されるオブジェクトの種類（棒グラフ、テーブル、領域グラフ、地図など）によってさまざまなバリエーションが数多くあります。下図7は、シンプルな一次元パターンの例で、URLへの総ヒット数を、サイトへ入って来たIPアドレス別の内訳を表示したものです。

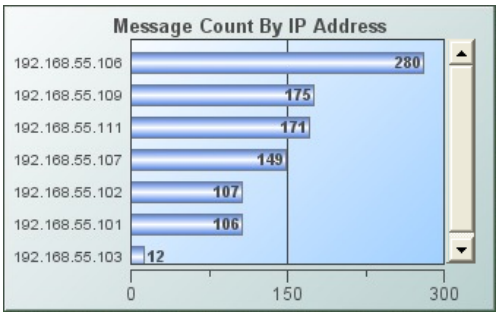

図7. IP アドレス別のメッセージ内訳

次の図8は、より多くの関わりを持った例になります。このシステムは、MMO（大規模多人数オンライン）ゲーム・アプリケーションにおけるアクティビティを監視するもので、ヒート・マップ上に最新のゲーム・プレイヤーとモンスターの位置が表示されています。ここで、アイコンの大きさは、それぞれのキャラクターが持つゲームにおける相対的な強さを表しています。

画面下方の棒グラフでは、プレイヤー数とモンスター数が、レース別とレベル別の内訳で表示されています。画面左方のパネルでは、闘士のアクティビティすべての時系列トレンドを、対話の種類別内訳で表示しています。ここで画面のグラフそれぞれは、入って来るデータに変換関数を適用した結果のテーブルに直接接続されています。興味深い点は、この可視化における生データ・セットは、リアルタイムなイベントに対して相関化処理をするCEPアプリケーションによる出力だという点です。

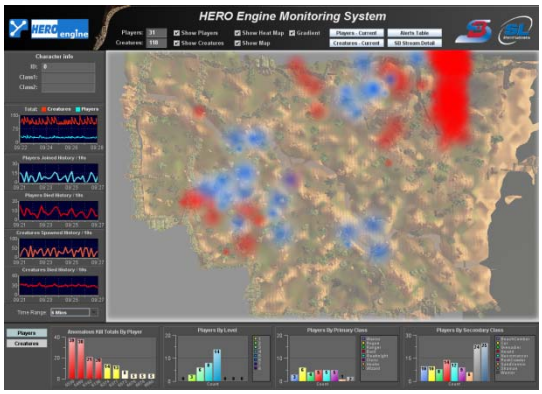

図8. プレイヤーとモンスターのレベルとアクティビティ別の内訳を表示するゲーム・アプリケーション

生データの集約と内訳表示（統計）の論理的な定義のすべては、画面の定義に含まれているという点に注目してください。関数自身は、データ・サーバ内で自動的に実行するように構成されており、アプリケーションを監視しているそれぞれのクライアント側における負荷を最小限にしています。

B. ベースライン・トレンド

リアルタイム・データの評価指標をトレンドグラフで表示することは、長年に渡って行われてきたことです。プロセス制御やテレメトリ・アプリケーションでは、これら複数のトレンドグラフをストリップチャートに表示する技術は昔に確立されていました。

エンタープライズ・クラスのBAMアプリケーションでは、リアルタイム・データのトレンド化において、これら従来のアプリケーションとは異なる点が2つあります。

一つ目は、トレンド化の性質そのものです。ほとんどのプロセス制御アプリケーションでは、プロセスは連続的で一日24時間中稼動しており、リアルタイム・データ値の監視は、これらがいつでも特定の範囲内であることを確保するためのものです。

一方BAMアプリケーションでは、監視されているリアルタイム・データは、一日24時間の中で大きく変動します。たとえば、ある一つのサーバで処理されるメッセージ数や、Webサイトへのログイン・ユーザ数は、一日のうちのある時間帯や一週間のうちの曜日によって、変動します。従って、現在のデータ値がその時間帯における典型的な範囲または「ベースライン」の外にないかどうかを、監視するものです。（ベースラインはフラットな線ではなく、一日の中で上がったり下がったりのサイクルがあります。）

次の図９では、Webサイトにログインするユーザ数のトレンドを、「チャネル」というベースラインに対してプロットしたものです。つまり、一日における時間帯と一週間における曜日によるトレンドを計らい、直近4週間平均データの上下30%範囲を取ったものです。

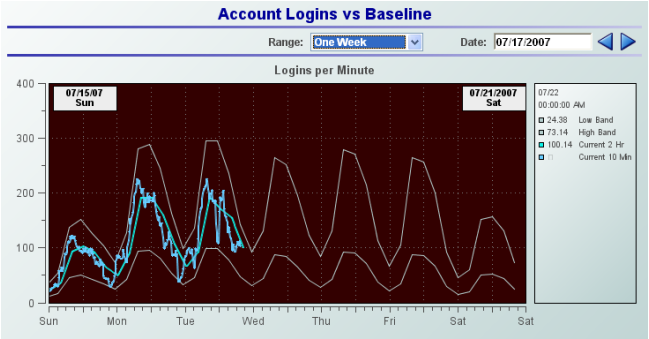

図9. ベースライン・トレンド比較の例

ベースライン・チャネルの使用は、ビジネスに関連したリアルタイム・データの可視化で、最も重要で有効な手法の一つです。監視している値がこのベースライン・チャネルの外へ出たときにだけ、アラートを発生するように構成できるからです。

リアルタイムなビジネス・データで異なる二つ目の点は、トレンドグラフの次元軸に沿ってデータを集約することが重要だということです。たとえば、注文を処理しているサーバが8台あり、最も良い応答時間をユーザに提供しようとシステムを解析しているとします。このケースでは、全サーバ8台に送信されるすべてのメッセージ全部が重要であって、サーバ1台におけるメッセージ数ではありません。異なる次元に沿ったデータ集約を自動化し、ユーザが全サーバ8台または特定のサーバ1台を選択できることが、どのようなビジネスに関連した監視システムにおいても、重要な機能になります。

VIII. 結論

さまざまなタイプの監視システムにおけるリアルタイム・データの取得は、歴史的に進化してきました。

従来のプロセス監視制御システムが主にポイント・ベースのデータを扱ってきたのに対し、ビジネス・アクティビティ・モニタリング(BAM)アプリケーションでは、メッセージング・ミドルウェアやその他の構造化されたリアルタイムなデータソースから取得し、これらデータの解析とアーカイブが複雑化しています。しかしながら、両システムにおいて、多くの可視化と解析における要求条件は類似しています。

従来のプロセス監視制御システムからの教訓は、数々のミッション・クリティカルなアプリケーションで実証されてきたものであり、貴重であると言えます。無論、BAM アプリケーション分野には新たなる要求条件に対応した技術が必要ですが、従来のリアルタイム監視制御技術のベースがあるか否かが大きな違いをもたらします。

ときにして、ビジネス・アクティビティ・モニタリング(BAM)アプリケーションを自社開発しようという試みが見られることもありますが、要求される機能の広範さと深さに、注意が必要です。データをいかに迅速に解析してプレゼンテーションするかを軽視したアプリケーションの開発・保守・拡張は、極めて困難でコスト高なものになります。

リアルタイム・データのインメモリにおける永続化とデータベース・アーカイブは進化しています。現在のシステムは標準的なリレーショナル・データベースを使っていますが、徐々に多次元のリアルタイム・データ処理を目的としたシステムへと移って来ています。また、レガシーなデータベース・システムが、より高速なリアルタイム・データを取り扱えるようにしたものもあります。しかしながら、これら従来のシステムにおけるリアルタイム・データの扱いには限界もあります。

そして、解析も標準的な SQL からストリーム処理の適用へと進化しています。さらに、時系列で多次元にキャッシュされたデータに対して操作を行なう変換関数の基本的なセットが、ほとんどの要求条件を満たすことが実証されています。

総じて、効果的な BAM では、リアルタイム・データは効率的に処理されなくてはいけません。そして、関連する情報をリアルタイム・データのストリームから抽出して解析し、さまざまなユーザに対して、Web ベースのダッシュボードやアラートによって、プレゼンテーションする必要があります。これらのリアルタイムでミッション・クリティカルなデータの操作においては、従来のプロセス制御における監視アプリケーションから、多くを学ぶことができます。

ただし、成功する BAM ソリューションの開発に伴う技法は、従来のプロセス制御における監視アプリケーションと類似する点が多くあるものの、技術革新も必要とします。BAM は、「エンタープライズにおけるプロセス制御」として捉えることができ、古い技術と新しい技術の組み合わせが、フル機能のビジネス・アクティビティ・モニタリング(BAM)アプリケーションにおけるリアルタイムなデータの解析と可視化の要求条件を満たすのに不可欠である、ということが分かります。

参考文献

[1] McCoy, David – Business Activity Monitoring, ID Number: LE-15-9727, 1 April 2002

[2] Gassman, Bill – Guide to Process-Centric BI Terms, Gartner Research, ID Number: G00151682, 21 September 2007

[3] Kemsley, Sandy – Blog on Technology in Business, http://www.intelligententerprise.com/blog/archives/2007/09/gartner_bpm_day_1.html, September 18, 2007

[4] Schulte, Roy - "The Business Impact of Event Processing: Why Mainstream Companies Will Soon Use A Lot More EDA,", SCW, p. 51, IEEE Services Computing Workshops (SCW'06), 2006

[5] Iwanitz, Frank and Lange, Jurgen – OPC – Fundamentals, Implementation, and Application, Huthig Fachverlag, 2006

[6] Defining Technology, Inc - Middleware Resource Center – www.middleware.org, 2008

[7] Luckham, David – The Power of Events: An Introduction to Complex Event Processing in Distributed Systems, Addison-Wesley, 2002

[8] SUN – Java Management Extensions (JMX) - Best Practices, 2007

[9] National Instruments – A Current Value Table for LABView, http://zone.ni.com/devzone/cda/tut/p/id/6108, 2008

[10] OISsoft - PI System Data Sheet, www.osisoft.com, 2008

[11] AspenTech – Real-Time Data Historian Data Sheet, www.aspentech.com, 2008

[12] Oetiker, Tobias – RRDtool Documentation, http://oss.oetiker.ch/rrdtool, 2008

株式会社 SL ジャパン
〒107-0062 東京都港区南青山3-8-5 アーバンプレム南青山 3階
Tel. 03-3423-6051 Fax. 03-3423-6070 info@sl-j.co.jp www.sl-j.co.jp