Smarter Citiesの実現に向けたストリーム・コンピューティングとイベント相関分析エンジンの連携によるリアルタイム・イベント処理

田代 孝仁　後藤 泰久　古郷 誠　福井 祐人

Real-Time Event Processing through Integrating Stream Computing and the Event Correlation Engine for Smarter Cities

Takahito Tashiro, Yasuhisa Gotoh, Makoto Kogoh and Yuto Fukui

Smarter Cities は，都市における人間活動を支えるあらゆるものをITによって接続・管理し，それぞれの動作を協調させ最適化することを目指すものである．この実現に不可欠となる大量イベント処理のリアルタイム性と，イベント相関分析エンジンによる高度なイベント処理を両立させるために，InfoSphere StreamsとTivoli Netcoolをイベント変換アダプターにより連携させる手法を提案する．この適用事例として携帯電話網の障害検知のためのプロトタイプを開発，性能評価を行い，実装時の考慮点を明らかにした．

We developed a general-purpose event conversion adapter to integrate InfoSphere Streams with Tivoli Netcool. We then built a prototype system that enables the real-time, advanced event processing required to realize Smarter Cities. This paper presents the details of the aforementioned prototype system, reflects upon its development, shows a potential use case for clients, and details the results of a performance examination of the system.

Key Words & Phrases：Smarter Cities, ストリーム・コンピューティング，リアルタイム・イベント処理，イベント相関分析，障害検知
Smarter Cities, Stream Computing, Real-Time Event Processing, Event Correlation, Fault Detection

1．はじめに

Smarter Citiesは，2008年よりIBMが提唱しているSmarter Planet［1］の構成要素の1つであり，都市における人間活動を支えるあらゆるものをITによって接続・管理し，それぞれの動作を協調させ最適化することを目指すものである．ITを活用し，エネルギー，水，通信などの生活インフラの効率利用，交通渋滞の緩和など，都市における問題を解決することを目的とする［2］．

Smarter Citiesでは，無数の機器が管理対象となり，それらを監視するセンサーにより膨大な数のイベントが絶え間なく検知される．都市では，あらゆる人・物が相互に作用しながら独立して動作しており，それらの状況は刻々と変化する．そのため，都市全体の機能を最適化するためには，集約されたイベントをリアルタイムに処理し，個々の構成要素だけではなく関連する要素の状態や相関を把握した上で，全体としての機能が最適となるよう各要素の次の動作を決定・実施させることが必要となる．

提出日：2010年9月6日　再提出日：2011年9月6日

Smarter Citiesを含むSmarter Planetに求められるリアルタイム性の高いイベント処理を実現するための基盤技術の1つとして，ストリーム・コンピューティングが注目されている［3］［4］．ストリーム・コンピューティングは，多様かつ互換性のない複数の情報源から連続的に生成されるデータ（ストリーム・データ）を，ストレージに保管することなく，リアルタイムに処理することを目的としたコンピューティング・モデルである．IBMは，ストリーム・コンピューティングの開発・実行のためのプラットフォームであるInfoSphere Streams（以下，Streams）を開発し，2010年に製品として発表した［5］．

Streamsは，独自に開発されたプログラミング言語SPL（Streams Processing Language）［6］を提供しており，基本的なデータ処理オペレーターを組み合わせることで，ストリーム・コンピューティングのロジックを直観的に記述することができる［7］．しかし，数千万・数億規模の人・物が稼働する都市の膨大なイベントの中から相関を見つけ出し，最適な動作を決定するためには高度なイベント分析ロジックが必要となり，SPLが提供する基本オペレーターを組み合わせて最初から実装していくのは，開発効率の観点から現実的ではない．

図 1. InfoSphere Streams と Tivoli 製品の連携による Smarter Cities の実現

筆者らは，Smarter Cities の実現に不可欠となる大量イベント処理のリアルタイム性と，イベント相関分析エンジンによる高度なイベント処理機能を両立させるために，IBM 製品の InfoSphere Streams と Tivoli Netcool（以下，Netcool）を連携させる手法を提案する．Streams により膨大な数のイベントの中から対応が必要な重要イベントのみを抽出して Netcool に送り，複合的な要素から適切な処理を判断し実行する．あらかじめ処理が必要なイベントのみがイベント相関分析エンジンに送られるため，Netcool では計算リソースをより高度な処理に集中させることができる．

これら両製品を連携させるために，イベント変換アダプターを開発した（図 1）．この適用事例として，携帯電話網の障害検知のためのプロトタイプを開発し，その性能評価を行った．

2. Smarter Cities 実現のための技術とその課題

2.1 ストリーム・コンピューティングによる 高速イベント処理

ストリーム・データの処理プログラムを記述する SPL では，プリミティブな機能を提供する基本オペレーター，ユーザーがその動作を自由に実装できる UDOP（User-Defined Operator）と UBOP（User-defined Built-in Operator），データベースへの入出力用のオペレーターなどを提供しており，これらを組み合わせてプログラミングする．1 オペレーターが 1 処理に相当し，処理の流れに即してオペレーターを配置・接続することでプログラムを作成できるため実装が容易であり，プログラムの可読性も高い．

また，スケーラビリティーの高さも SPL の特徴の 1 つである．各オペレーターを別マシン上で実行させるよう再配置することにより，常に拡大を続ける都市のデータ量に合わせ，システムの増強が必要となる場合にも柔軟に対応できる．

一方，SPL ではプリミティブなオペレーターを組み合わせることで，柔軟性の高いプログラム実装が実現できる反面，すべての処理ロジックをユーザーが独自に組み上げることが求められる．このため，複雑かつ高度なイベント処理が必要な場合には，開発時の負荷が非常に大きくなる．

2.2 イベント相関分析エンジンによる高度なイベント処理

Netcool は，大規模環境におけるシステム監視を実現する製品群である．イベント・エンジンである Netcool/OMNIbus を中心に，複数の監視対象の統合監視を可能とするほか，監視結果を一元的に管理，分析する機能を提供しており，より上位レベルのシステム監視を実現する．また，イベント相関分析エンジンである Netcool/Impact を用いることで，イベントへの各種情報の付与，相関分析などが可能になり，高度なイベント処理を実現することができる．

例えば複数のイベントが発生した場合，ナレッジ・データベースやネットワーク・トポロジーなどを参照しながら，イベント同士の相関分析を行うことで，主原因イベントと，それにより生じた副次イベントに分類することが可能になる．

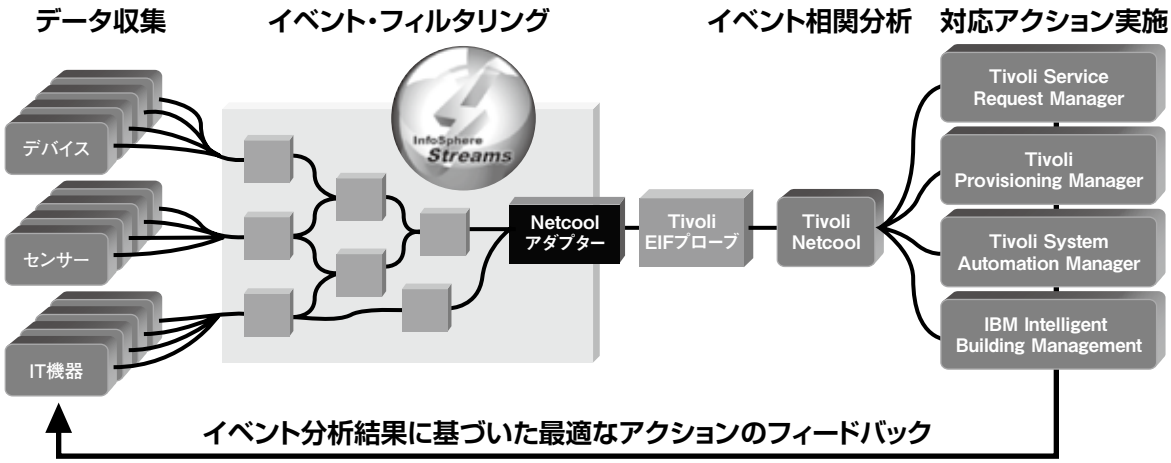

図 2. InfoSphere Streams と Tivoli Netcool の連携によるリアルタイム・イベント処理システム

さらに構成情報データベースなどから障害の発生した機器の管理者情報を取得し，管理者に対して通知を送ることで，より迅速な問題解決が可能になる．このように，さまざまなデータ・ソースからの情報を正規化し，イベント相関分析エンジンを活用することで，単純な IT レベルでのイベント監視だけではなく，サービス・レベル，ビジネス・レベルといった多階層での監視を実現できるようになる．

Netcool では多様なデータ・ソースからのデータ収集を実現するために，それぞれのデータ・ソースに特化した多数の専用のイベント抽出モジュールを提供している．Smarter Cities においては，都市に存在するあらゆるものがデータ・ソースとなりうるため，そのすべてに対して専用のイベント抽出モジュールを提供することは現実的ではない．すなわち，Netcool のイベント処理能力を Smarter Cities において十分に活かすためには，Smarter Cities で生み出される膨大な種類と数のデータから，真に解析が必要なイベントのみを的確に抽出し，Netcool の入力として橋渡しをするための仕組みが必要となる．

3. ストリーム・コンピューティングとイベント相関分析エンジンの連携

本論文では，図２に示すように高速なイベント処理を可能とする Streams と，イベント相関分析エンジンによる高度なイベント処理を可能とする Netcool を連携させるためにイベント変換アダプターを開発し，両製品を融合させたシステムを実現することを提案する．

まず，Smarter Cities から絶え間なく発生するデータを Streams で受け取り，高度な相関分析が必要なイベントの抽出を行う．抽出されたイベントは，イベント変換アダプターを経由して Streams から Netcool に送られる．Netcool はイベント相関分析エンジンを使用して受信したイベント群の相関分析を行い，イベントの原因分析や影響範囲の特定などの処理を行う．さらにその処理結果を，社会システムを管理するソフトウェア（IBM Tivoli Service Request Manager や IBM Intelligent Building Management など）に入力することで，イベント分析結果に基づいた最適な管理アクションを実施させることが可能になる．

3.1 イベント変換アダプターの開発

1） イベント変換形式

Streams と Netcool の連携には，Streams プログラムが処理したイベント・データを，Netcool が受信可能な形式に変換する必要がある．筆者らは，Streams プログラム上で利用できるイベント変換オペレーターの形式を持つイベント変換アダプターを新規に設計・開発した．

Netcool へ転送するためのイベント形式としては，Tivoli Event Integration Facility（EIF）[8] を採用した．EIF は Netcool をはじめ，Tivoli Enterprise Console や Tivoli Monitoring など，多くの Tivoli 製品でサポートされている汎用的なイベント形式である．本イベント変換アダプターを使用することで，Netcool 以外の他の Tivoli 製品も Streams との連携が可能になる．

2） データ・スキーマ構造定義ファイル

Streams におけるイベント変換オペレーター実装時の制約として，SPL 上で定義された入力・出力データのスキーマ構造に即した実装を行う必要がある．そのため，スキーマ構造が異なる場合には通常オペレーターの実装の変更が必要となるため,汎用性に乏しくなる．そこで筆者らは，スキーマ構造の差異を隠ぺいできるよう，汎用的な設計を行った．

スキーマ構造に依存しないイベント変換ロジックを実装

するためには，アダプターの内部からスキーマ構造を参照する仕組みが必要となる．これには 2 つの方法が考えられる．1 つ目の方法は，スキーマ構造を記述した定義ファイルを用意する方法である．イベント変換アダプターの起動時に定義ファイルを読み込ませることで，スキーマ構造をアダプターに知らせることができる．2 つ目の方法は，Streams が提供するオペレーター API を使用する方法である．オペレーター API には，入力データのスキーマ構造を取得するメソッドが用意されている．さらに，スキーマ内の各属性の型と，属性の名前もしくはインデックス（スキーマ内での属性の順番）を指定することで，各属性の値を取得できる．

今回開発したイベント変換アダプターでは，上記の 2 つの方法を組み合わせた手法を採用した．すなわち，データのスキーマ構造を記した定義ファイルを用意し，データの各属性の値はオペレーター API 経由で取得する手法とした．これは，変換されたイベントの受け取り手となる Netcool などの外部アプリケーションとの連携を考慮したものである．

Netcool でイベントを受信し，適切な処理を行おうとした場合，イベントのスキーマ構造に合わせた事前構成が必要となる．その際，通常は Streams が出力するイベントのスキーマ構造を知るために SPL プログラムの判読が必要となる．これに対し，出力イベントのスキーマ構造が定義ファイルに記述してある場合，その定義ファイルからスキーマ構造を参照できるため，Netcool 事前構成時の SPL プログラムの判読スキルが不要となる．

4. 想定適用事例 – 携帯電話網の障害検知

携帯電話加入契約数は，2010 年には世界全体で 50 億件に達する［9］と言われており，携帯電話網は既に都市の重要なインフラの 1 つとなっている．その結果，障害発生時の都市生活への影響は甚大であり，リアルタイムな障害検知，迅速な復旧作業の実施が必須となる．

筆者らは，Smarter Cities の一例として携帯電話網の障害検知の事例を想定し，Streams と Netcool の連携によるリアルタイム・イベント処理機能を活用した携帯電話網の障害検知のためのプロトタイプ・システムを開発し，その評価を行った．

4.1 想定シナリオ

携帯電話では 1 回の通信要求ごとに Call Detail Record（CDR，通話詳細記録）が 1 レコードずつ，携帯電話基地局上に生成される．一般的に CDR には，発着信者番号，発信・着信・切断日時，切断理由コードなどが含まれる．切断理由コードは，正常な終話のほか，電話網の障害や輻輳（ふくそう）など，通話が切断された理由を示している．CDR は利用者の従量課金額の計算に使われるほか，切断理由コードを解析することで携帯電話網の状況を知ることができる．

課金額計算の場合，課金単位期間に一度まとめて計算すればよいが，障害検知への利用ではリアルタイムでの解析と検知が求められる．例えば，1,000 万人の利用者がいる都市で，各利用者が 30 分に 1 回発信すると

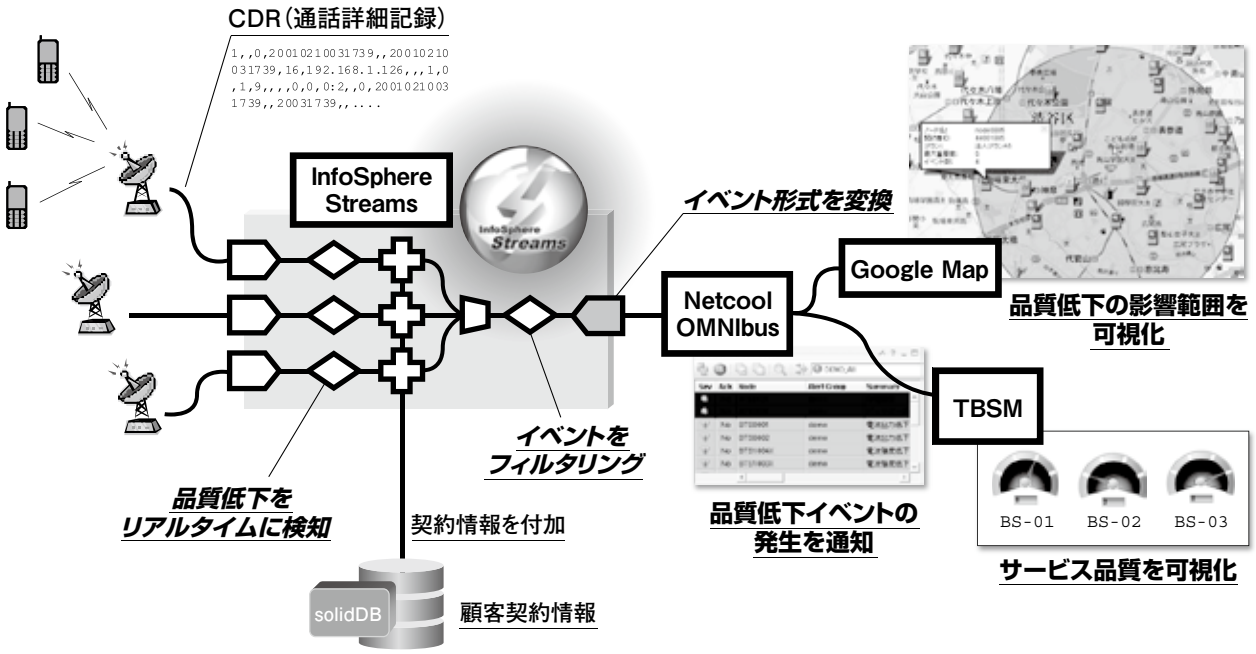

図 3. 携帯電話網の障害検知のためのプロトタイプ・システム

仮定すると，毎秒 5,500 件以上の CDR が生成され，これを遅延なく解析する必要がある．また，複数の障害が同時に発生した場合，個々の加入者の契約内容や Service Level Agreement （SLA）を考慮し，障害対応の優先度を設定する必要もある．

4.2 携帯電話網の障害検知システム構成

図 3 は，今回プロトタイプ・システムとして構築した携帯電話網の障害検知システムの概略である．基地局で生成される CDR を Streams で解析し，まずは障害の有無を検知する．障害を検知すると，顧客の契約情報データベースを参照し，該当する顧客の SLA に応じてその障害の重大度を設定する．Streams による障害検知処理の詳細については後述する．

検知された障害はイベント変換アダプターにより EIF 形式のイベントに変換され，Netcool に送られる．Netcool は，受け取ったイベントの一覧を表示し，さらに Tivoli Business Service Manager （TBSM） および Google Map にイベントを転送する．TBSM は，エンドユーザーに提供されるサービスの品質管理を行うためのソフトウェアであり，ここでは各基地局上で発生している障害の比率をメーターで表示し，サービス品質を可視化する．一方，Google Map では，障害が発生した携帯電話端末，基地局，およびその基地局の担当範囲の位置をアイコンと円で表示し，障害の影響範囲を可視化する．画面上のアイコンをクリックすることで，障害や端末の契約情報の詳細を確認することも可能である．

4.3 Streams による障害検知処理

図 4 に，Streams 上に実装した障害検知プログラムの処理フローを示す．基地局で生成される CDR は，CSV 形式により 1 分間隔で取り込まれ，データ構造が解析される．ここでデータは 2 つのストリームに複製され，一方は各携帯電話端末の障害検知処理，他方は基地局の障害検知処理に利用される．

携帯電話端末の障害検知処理では，端末ごとに CDR データを集計し，障害比率を算出する．次に，端末に該当する顧客の契約情報をデータベースから取得し，データに障害比率に関する SLA の値を付加する．データベースには，リアルタイム処理性能が高いインメモリー・データベースである solidDB を利用する．次に，算出された障害比率を SLA の値と比較することで，SLA を満たしていないものを抽出し，同時に問題の重大度を付加する．さらに，発生した障害に関する詳細情報を再度データベースから取得して付加する．最後に，イベント変換アダプターで EIF イベントに変換し Netcool/OMNIbus に送付する．

基地局の障害検知処理においても同様に基地局ごと

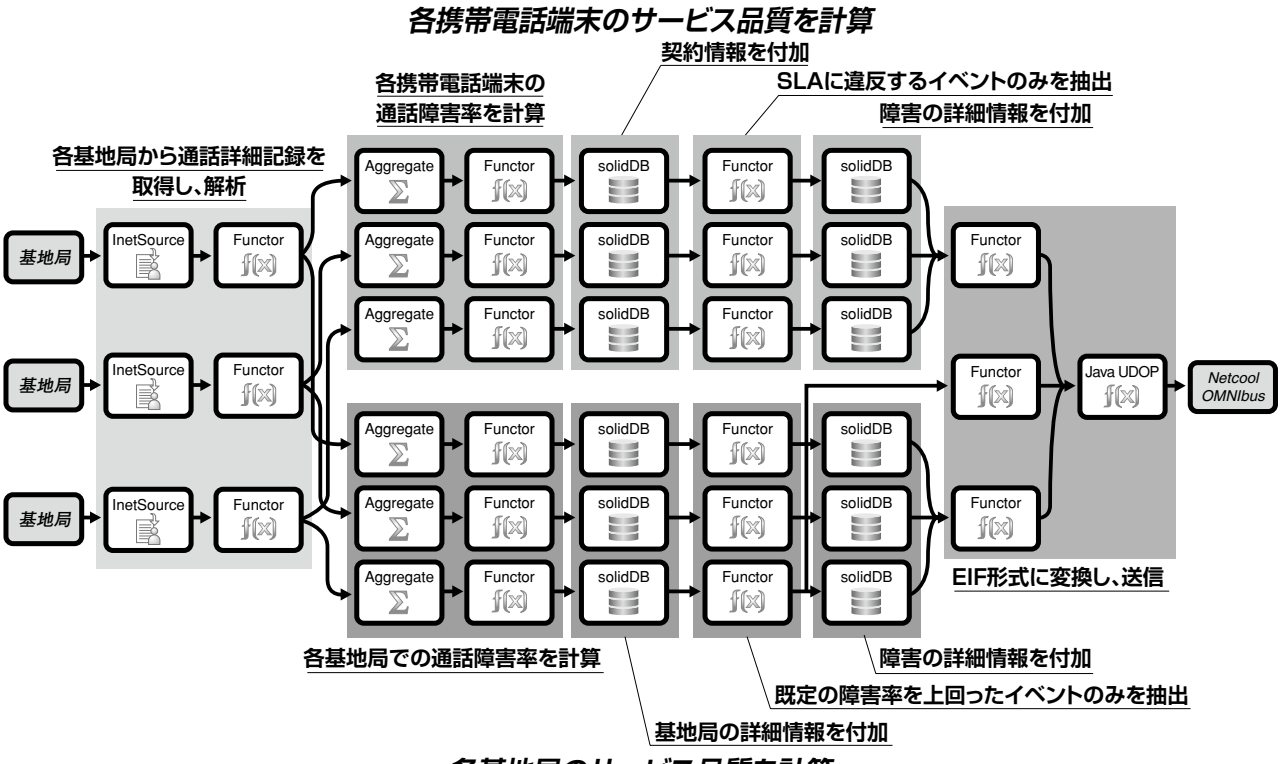

図 4. InfoSphere Streams で実装した携帯電話網の障害検知プログラムの処理フロー

の障害比率を算出し，データベースを用いて基地局の詳細情報を取得する．ここで，さらにストリームを 2 つに複製し，一方はそのまま EIF 形式に変換して Netcool に送り，TBSM 上で基地局ごとの障害比率を可視化するために利用する．他方は，障害イベントとして表示するため，障害比率が一定値を超えた場合のみ，障害に関する詳細情報を付加してから EIF イベントに変換し，Netcool/OMNIbus に送信する．

4.4 適用結果

本プロトタイプ・システムを用いて，3 つの基地局の担当範囲内に 30 万の利用者がいることを想定したシミュレーションを行った．CDR データは，シミュレーターによって生成した．30 万台の携帯電話モデルが無作為な間隔で発信し，1 分間に 20 万件程度の CDR を生成させた．各通話にはランダムな比率で障害を発生させ，また，各利用者には異なる SLA を設定し，障害の重大度が利用者ごとに変わるようにした．

このケースでは，各携帯電話，基地局の障害が期待通り検知され，SLA に応じた重大度が割り当てられた上で Netcool に通知された．Streams のデータ取得間隔と Netcool の GUI の更新間隔が共に 1 分のため，障害発生から Netcool 上に障害が表示されるまでに最大 2 分程度の時間差が生じたが，それ以上の大きな遅延は見られなかった．なお，本システムは第 5 章の性能評価実験で使用した仮想マシン上に構築した．

本システムを実環境へ適用する場合，一連のオペレーター・セットを基地局の数に合わせて追加することでスケーラビリティーを確保することが可能である．また，データ量の増加により処理パフォーマンスが低下した場合，負荷が高いオペレーターを別ノードに振り分けることで対応することが可能である．

5. イベント変換アダプターの性能評価

Streams によるイベント抽出処理，および開発したイベント変換アダプターの性能評価のため，評価用 Streams プログラムを実装し，処理パフォーマンスを測定した．評価用プログラムは，複数のイベントの集約結果から問題を検知し，Netcool へ通知することを想定して実装した．

プログラムのパフォーマンスは，1 イベント当たりの処理時間と，1 秒間に処理できるイベント数（スループット）の 2 つの観点で評価した．プログラムに一度に入力するイベント数を変え，その違いによる処理時間とスループットの変化を調べた．

5.1 性能評価手順

イベント・データが Streams プログラムに取り込まれてから EIF イベントに変換されてイベント変換アダプターから出力されるまでの時間の平均値を計測し，これを処理時間とした．また，一度に入力されたイベント群の処理に要する平均時間から，1 秒間に処理されたイベント数を求め，これをスループットとした．

一度に入力する数のイベントを 1 つの CSV （Comma Separated Values）ファイルとして用意し，ファイル内のすべてのデータを TCP（Transmission Control Protocol）経由で Streams プログラムに入力した．ファイル内のすべてのデータの送信が完了するまでを 1 回の操作とし，この操作を連続的に 1,000 回繰り返して測定を行った．CSV ファイルの 1 行が 1 個のイベント･データを表しており，また，各イベント・データは，イベントを一意に識別するキー文字列と，1 つの実数値を持つ．ファイル内のイベントはすべて異なるキー文字列を持ち，別イベントとして認識される．

実験プログラムでは，同じキーを持つイベントを 10 回分集約し，そのイベントの実数値の総和が一定のしきい値以上のものを抽出する．その後，抽出されたイベントを，イベント変換アダプターにより EIF イベントに変換する．

実験には，VMware ESX Server 4.0 上に作成した仮想マシン 1 台（クロック周波数 3GHz の 32 ビット CPU × 2，メモリー容量 2GB）を利用した．OS には Red Hat Enterprise Linux 5.5 を用いた．

5.2 評価結果

図 5 に実験結果を示す．横軸は一度に入力されるイベント数，左縦軸はイベント変換アダプターが 1 秒間に処理可能なイベント数（スループット）の平均値，右縦軸はプログラムが 1 イベントの処理に要する平均処理時間を表す．

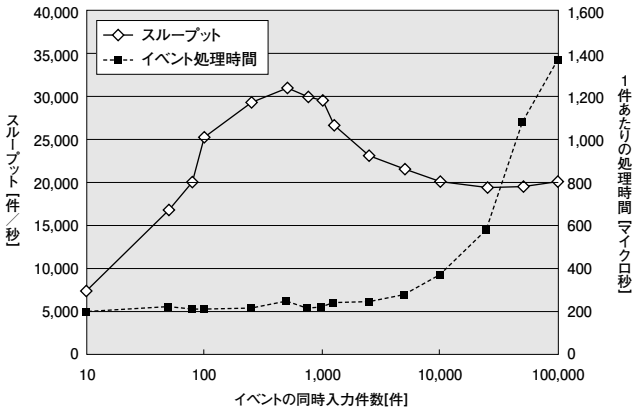

図 5. イベントの同時入力数の相違よる Streams プログラムの処理スループットとイベント 1 件あたりの処理時間の変化

実験結果では，処理スループットはイベントの同時入力数が 500 件の場合に最大となり，秒間約 31,000 件であった．さらに同時入力件数を増やすとスループットは低下し，10,000 件以上の入力では，秒間 20,000 件程度の値に収束した．

1 イベント当たりの処理時間は，同時入力件数が 5,000 件程度までは，1 イベント当たり 200 マイクロ秒前後であった．同時入力件数が 10,000 件を超えると，1イベント当たりの処理時間は増加し，10,000 件の入力では 369 マイクロ秒，50,000 件では 1,083 マイクロ秒，100,000 件では 1,370 マイクロ秒となった．

5.3 考察

本実験では，同時入力イベント件数が 10 件から 500 件までは，件数が多くなるにつれてスループットが高くなる結果が得られた．これは，イベントの同時入力件数が少ない場合，イベントが TCP 経由で Streams に入力される際の TCP プロトコル処理のオーバーヘッドが相対的に大きいことに起因していると考えられる．

本実験で計測した 1 イベント当たりの処理時間は，Streams プログラムに取り込まれたイベントの集計・変換の所要時間のみを計測しており，イベントの Streams への入力処理の所要時間は含めていない．しかし，Streams の特性上，イベントの入力処理と集計・変換処理は同時並行で逐次的に行われるため，一方の処理で多くの計算リソースが消費されると，他方の処理性能に影響を与える．イベントの Streams への入力処理の内訳は，大きく TCP プロトコル処理とバッファリング処理に分けられるが，TCP プロトコル処理にはデータ量にかかわらず一定の処理時間がかかる．そのため，データ量が小さい場合には TCP プロトコルの処理時間が大半を占め，比較的負荷の小さいバッファリング処理の時間が少なくなる．この結果，イベントの集計・変換処理中には，ほぼ TCP プロトコル処理が並行して行われていることになり，その影響で利用可能な計算リソースが相対的に少なくなると考えられる．処理するイベントの量が多くなるにつれ，バッファリング処理が占める時間が大きくなることで，イベント集計・変換処理が利用する計算リソースの量が相対的に増加するため，結果的に処理のスループットが高くなったと考えられる．

一方，イベントの同時入力件数が 500 件を超えると，入力件数の増加に伴ってスループットが低くなり，さらに 10,000 件以降になるとスループットが収束した．この結果はそれぞれ，イベント入力時のバッファリング性能と，イベント集計・変換処理時の CPU 性能の限界値に関連があると考えられる．

スループットが下がる部分では，一度に入力されたイベントのバッファリング処理に要する時間が，集計・変換処理の所要時間よりも大きくなることで，集計・変換処理にイベントの入力待ち時間が含まれるようになったためである．TCP でのバッファリング処理では，一定量のイベントをまとめてバッファに格納するため，イベント量の増加が線形的に処理時間に影響されない．そのため，スループットの低下率は，データ量の増加率に比べ緩やかになる．

スループットが収束する部分では，同時に 1 イベントあたりの処理時間が急増している．すなわち，入力イベント量がCPUの処理性能の限界を上回ったことで,スループット性能が CPU 処理能力の限界点で飽和し，それに伴って処理待ちのイベント量が増加することで，イベントあたりの処理時間も線形的に増加していることが読み取れる．

以上の結果から，Streams プログラムに対する入力イベント量が多くなった場合，一定の処理スループットを期待することができるが，イベントの増加量が CPU 性能の処理限界を超えると，1 イベントあたりの処理時間の増大につながると考えられる．このことから，発生イベント数が大きくなることが予想されるケースでは，処理ノード数を増やす形で処理負荷の分散を検討すべきであることが分かる．

6．おわりに

Smarter Cities の実現に不可欠となる大量イベントのリアルタイム処理が，ストリーム・コンピューティングとイベント相関分析エンジンの連携によって実現でき，イベント処理のリアルタイム性と高度なイベント処理を両立するシステムが構築可能であることを示した．今回は想定適用事例として，携帯電話網の障害検知のためのプロトタイプ・システムを開発し，シミュレーションによって実現可能性を示した．

Smarter Cities の実現には，収集された膨大な数のイベントから対処すべきイベントを検知し，さらにそのイベントに関係する人・物の状態，およびそれらの関連を的確に把握した上で，イベントに対処する最適なアクションを決定・実施することが必要となる．例えば，Tivoli Maximo などのような資産管理ソフトウェアと連携し，イベントの解析結果から得られた機器や資産の稼働状況，使用実績などを利用することで，これまで経過期間などで画一的に実施されていた保全作業を，より効果的かつ効率的なタイミングで実施できるようになる．さらに，各作業員の現在位置と周辺地域の交通状況をイベントとして解析し，

障害発生地点へ最も早く到達可能な作業員に復旧作業を割り当てるといった，状況に即した対応アクションを行うことも可能になる．

このように，都市から収集されるイベントを解析し，その結果をアクションとしてフィードバックすることで，データ収集・監視・可視化・管理・アクション実行という改善サイクルを確立することができる．実施されたアクションの影響も，種々のセンサーやデバイスなどにより即座にイベントとして収集・解析され，継続的に改善が行われていく．このような継続的な改善サイクルを都市に組み込むことで，初めて Smarter Cities を実現することができる．

今後も都市全体の改善サイクルの管理を目的として，サイクルの各フェーズを担当するコンポーネントを有機的に連携させ，Smarter Cities を具現化していきたい．

謝辞

本論文の執筆にあたり，東京基礎研究所 インフラストラクチャ・ソフトウェア 古関聰氏，鈴村豊太郎氏，小野寺民也氏には，大変有益なご助言をいただきました．あらためて深謝いたします．

参考文献

[1] 久世和資: "Smarter Planet―イノベーションが実現する社会の未来価値," ProVISION, No.64, pp.6-11 (2010).
[2] 志済聡子: "地球を「スマート」にするには,まず都市から," ProVISION, No.64, pp.12-17 (2010).
[3] Daniel J. Abadi, et.al: "The Design of the BorealisStream Processing Engine," 2nd Biennial Conference on Innovative Data Systems Research (CIDR 2005) (2005).
[4] Bugra Gedik, et.al: "SPADE: The System S Declarative Stream Processing Engine," the 2010 ACM SIGMOD/PODS Conference (2008).
[5] IBM Press Release: "IBM Ushers In Era Of Stream Computing," http://www.ibm.com/press/us/en/pressrelease/27508.wss (2009).
[6] IBM: "IBM InfoSphere Streams Version 1.2 Programming Model and Language Reference," http://publib.boulder.ibm.com/infocenter/streams/v1r2/topic/com.ibm.swg.im.infosphere.streams.product.doc/doc/IBMInfoSphereStreams-LangRef.pdf (2010).
[7] 小野寺民也,安江俊明,鈴村豊太郎: "ストリーム・コンピューティング時代を開く基盤ソフトウェアIBM InfoSphere Streams," ProVISION, No.65, pp.39-44 (2010).
[8] IBM: "Tivoli Event Integration Facility Reference ," http://publib.boulder.ibm.com/tividd/td/tec/SC32-1241-00/en_US/PDF/ecoemst.pdf (2003).
[9] International Telecommunication Union Press Release: "ITU sees 5 billion mobile subscriptions globally in 2010," http://www.itu.int/net/pressoffice/press_releases/2010/06.aspx (2010).

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
Tivoli 開発
ソフトウェア・エンジニア

田代 孝仁 Takahito Tashiro

[プロフィール]
2007 年，日本 IBM 入社．以来，ソフトウェア開発研究所にて，オートノミック・コンピューティング関連ソフトウェア，大規模分散システム管理ソフトウェア，Smarter Planet 関連ソフトウェアの開発に従事．博士（情報科学）
takahito@jp.ibm.com

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
Tivoli 開発
アドバイザリー・ソフトウェア・エンジニア

後藤 泰久 Yasuhisa Gotoh

[プロフィール]
1989 年，日本 IBM 入社．通信システム関連ハードウェア開発，各種アプリケーション・ソフトウェア開発，システム管理ミドルウェア製品開発，オートノミック・コンピューティング関連ソフトウェア開発に従事．現在はソフトウェア開発研究所にてシステム管理ミドルウェア製品開発および Smarter Planet 関連ソフトウェア開発を担当．
gotohy@jp.ibm.com

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
Tivoli 開発
ソフトウェア・エンジニア

古郷 誠 Makoto Kogoh

[プロフィール]
1999 年，日本 IBM 入社．主任ソフトウェア開発エンジニア．ワークフロー・システムの開発，ハイパフォーマンス・コンピューティング等のプロジェクトに従事し，現在はネットワークを中心とした監視システムを担当．
makotox@jp.ibm.com

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
Tivoli 開発
ソフトウェア・エンジニア

福井 祐人 Yuto Fukui

[プロフィール]
2009 年，日本 IBM 入社．同社大和ソフトウェア開発研究所にて，分散システム管理ソフトウェアのテクニカル・サポート業務に従事．
yfukui@jp.ibm.com