社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005−UBI−9 (2)
2005／11／1

大規模化可能なコンテキストプラットフォームの提案

佐藤 正　磯山 和彦　吉田 万貴子

日本電気株式会社 〒108-8557 東京都港区芝浦 2-11-5

E-mail: t-satou@ig.jp.nec.com, k-iso@bc.jp.nec.com, m-yoshida@em.jp.nec.com

あらまし RFID，センサの普及に伴い，それらの情報源から集められる情報(イベント)の数は膨大になると考えられる．本稿では，これらのイベントから利用者が要求するイベントを抽出するコンテキストプラットフォームを提案する．提案方式では，入力される情報(イベント)が利用者の要求する条件（ルール）に一致するかどうかを確認する処理(イベント処理)を分散化することにより大規模化を可能とする．イベント処理の分散化方式が有効であることを検証し，さらに実機を用いた評価により，提案方式がイベント数にスケールことを確認した．ルールが 1000 個設定される場合，イベント処理のマシンを 1 台から 5 台に増加することによりスループットを 5.69 倍とすることができた．

キーワード コンテキスト，プラットフォーム，分散化，ルール・エンジン

Proposal of Scalable Context Platform

Tadashi SATO　Kazuhiko ISOYAMA　and　Makiko YOSHIDA

NEC Corporation　2-11-5 Shibaura, Minato-ku, Tokyo, 108-8557 Japan

E-mail: t-satou@ig.jp.nec.com, k-iso@bc.jp.nec.com, m-yoshida@em.jp.nec.com

Abstract The data generated from the sources like RFID readers and sensor devices become increased as those devices increase. In this paper, we propose the context platform which extracts the events required by the users of the platform from a huge number of events. The proposed method distributes the process which checks whether events match to the conditions set by the user so that the platform scales to the event increase. We confirm that the proposed distribution method effectively works. We implement the proposed system and evaluate the system. It is shown that the throughput of 5 processors system is 5.69 times higher than that of 1 processor system when 1000 rules are set.

Keyword Context, Platform, Distributed Processing, Rule Engine

1. はじめに

近年，様々なコンテキストアウェアサービスが提案されている[1]．このようなサービスでは，様々なデバイスからヒト・モノ・コトの状態を示すコンテキストを収集し，IM のメンバーシップ管理，歩行者ナビゲーション等のサービスに利用する．

実世界の情報を電子的に取り込み，それらの情報をさまざまなサービスへ応用することを目的として，RFID (Radio Frequency Identification)，センサが注目を集めている．RFID，センサの普及に伴い，それらの情報源から集められる情報量は膨大になると考えられる．また，これまでにない情報を取得できるようになることによって，それらの情報を利用するアプリケーションも拡大すると考えられる．

このようなデバイスから発生する情報（イベント）を利用し柔軟なサービスを実現する技術として Rapide[2]，GEM[3]，コトミエ[4]等のイベント処理が提案されている．このような技術では，イベント処理ルールによって，デバイスから発生する複数の下位レベルのイベントからヒト・モノ・コトのコンテキストを表す上位レベルのイベントを導き出すことができる．

しかし，このような技術では複雑なイベント処理は実現できるものの，複雑なルール記述，ステート管理が必要となるため，上記のようなサービスが普及した時にデバイスから発生する膨大なイベントを処理することは困難である．

そこで本稿では大量のイベントを処理し，かつ柔軟なコンテキストアウェアサービスを実現するためのコンテキストプラットフォームを提案する．図 1 のように，本コンテキストプラットフォームは，従来から提案されている一般的なコンテキストアウェアサービスのプラットフォームの下位層として働くことを想定している．そして，比較的シンプルなイベント処理ルールを採用し，数万～百万イベント/秒のイベントのフィルタリング・アグリゲーションを行い，上位コンテキストアウェアサービスプラットフォームが処理可能な

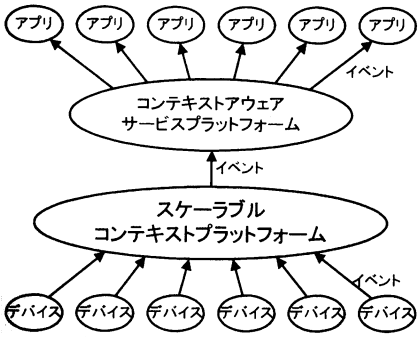

図 1 本コンテキスト PF の位置づけ

数百～数万イベント/秒までイベント量を絞って上位層に通知することを目的とする．以下では本コンテキストプラットフォームのアーキテクチャ，採用ルール形式，イベント処理の分散化方式および性能評価結果を述べる．

2. 提案方式

提案するコンテキストプラットフォームは上位層のコンテキストアウェアサービスプラットフォームから処理ルールを受信し，そのルールに従いデバイスからのイベントを処理し，上位層に結果を通知する．本プラットフォームでは，複数のイベントによって上位層に通知するか否かが決まるようなイベント処理(複合イベント処理)をサポートしている．これにより，デバイスからのイベントを効果的に絞り込める．

図 2 に提案プラットフォームの構成を示す．本プラットフォームでは，複数のイベントプロセッサ(EP)を並列に配置し，ディスパッチャにより各 EP が処理すべきイベントをその EP へ転送する方式を採用する．複合イベント処理を行うためにイベントのステート管理が必要となるが，EP 毎に独立に管理する方式をとる．

以下に本プラットフォームの動作概要を示す．

1) ルール設定フェーズ

EP 制御部(EP-CTL)は，上位層からアプリケーションルールを受け取ると，最適なルール分配を計算し，各 EP にルールを分配する．また，同時に EP-CTL は，それに連動して，ルールを設定された EP にそのルールの処理に必要なイベントが転送されるように，ディスパッチャにディスパッチルールを設定する．

このとき，各 EP には処理量が均等になるようにイベント処理ルールを分配する必要がある．しかし，無作為にルールを分配すると，多くの EP が同一のイベントを処理する必要が発生し，ディスパッチャにおいてイベントをコピーしそれら EP に分配する処理が発生するおそれがある．これを避けるためには同じイベントに関するルールは同じ EP に分配されるのが望ましい．これらを両立するルール分配アルゴリズムについては 2.2 で詳述する．

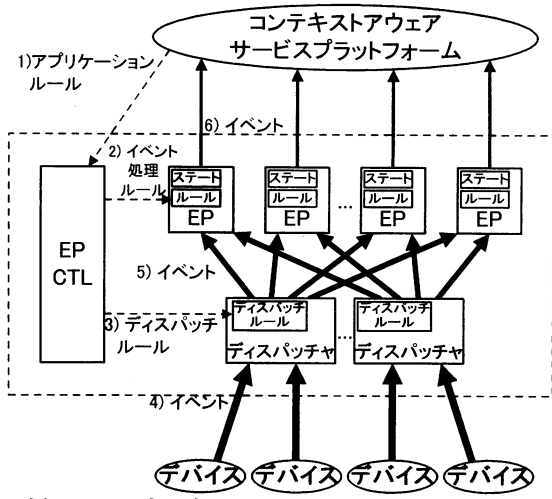

図2 コンテキスト PF アーキテクチャ

2) イベント処理フェーズ

デバイスから送出されたイベントはディスパッチャにおいてディスパッチルールに従って最適な EP に転送される．このとき，ディスパッチャはどの EP も必要としないイベントをフィルタアウトするため，EP が受信するイベント量が削減され，EP の負荷低減にもなる．そして，イベントを受信した EP は 1)で設定されたルールに従い複合イベント処理を行い，処理結果を上位層に通知する．

2.1. ルール形式

以下に採用するルール形式を示す．[2][3]で複合イベント処理ルールが提案されているが，我々はイベント処理の高速化とアプリケーションへの適用性のバランスを考慮し，そのサブセットを採用する．

1) Rule (Event A , Action X)
2) Rule (Event A | Event B , Action X)
3) Rule (Event A & Event B within t , Action X)
4) Rule (Event A -> Event B within t , Action X)

1)は単一イベント処理ルールである．2)～4)は複合イベント処理ルールであり，コンディションとして，2)は複数のイベントの OR，3)は AND を表す．4)は Event A の後に Event B が起きることを表す．また，“within t”はタイムオペレータであり，それら複数のイベントが時間 t 以内に起きることを表す．タイムオペレータによりステートを保持する時間を明確かつ有限に指定することにより，複合イベント処理で必要なステート管理を簡易にすることができ，高速処理が可能となる．

ディスパッチャにおけるディスパッチルールにも1)の単一イベント処理ルールを採用し，コンディションとしては単一属性値のみを指定する．これによりディスパッチャは単一イベントの単一属性値のみを見てイベントを転送するため，高速なイベント転送が可能となる．

2.2. ルール分配

2章で挙げた課題を解決するために以下の2つのポリシーに従ってルールを分配する．

1. 同じイベントを要求するEPを少なくする．
2. 各EPのイベント処理負荷のばらつきを一定以内に収める．

ポリシー1によって，ディスパッチャがイベントを転送する宛先であるEPの数を少なくでき，かつ，同じイベントを保持するEPを少なくできる．ポリシー2によって，特定のEPに高負荷がかかることを防ぐ．これら2つのポリシーはトレードオフの関係にある．例えば，ポリシー1に従うために全ルールを1つのEPに割り当てると，ポリシー2に従わなくなる．逆に，ポリシー2に従うために全EPに均等にルールを割り当てるとポリシー1に従わなくなる．以降，これら2つのポリシーに従うアルゴリズムを説明する．

ルールがAPから設定される度に以下の3つのステップを実施し，そのルールが割り当てられるEPを決定する．以下の各ステップにおいて候補を絞り，最終的に残ったEPにルールを割り当てる．

第1ステップでは，システム内の全EPの中からその時点で保持している最小のルール数との差が一定値(許容ルール数差)以下であるEPを選択する．

第2ステップでは，第1ステップで選択された集合からルールが要求するイベント群とEPが要求するイベント群で共通に要求するイベント群の数が最も多いEPを選択する．

第3ステップでは，第2ステップで選択された集合から要求する属性値の数が最も少ないEPを選択する．

以上のいずれかのステップでEPが1つに絞れた場合には，そのEPにルールを割り当てる．3つのステップ後に2つ以上のEPが残っていれば，その中からランダムに選択する．

3. 評価

3.1. ルール分配アルゴリズムの評価

提案アルゴリズムの比較対象としてラウンドロビン方式を用いる．評価に用いるルールに関して，歩行者ITSモデルとランダムモデルを用いる．

歩行者ITSモデルでは，いくつかの歩行者のグループがあり，同じグループ内のメンバー同士が近づいたら，メンバーに通知するルールを設定する．ルール形式は2.1節の3)となる。ランダムルールモデルのルール形式も上記と同様とし，ただし，ルール内のコンディションで指定する人と場所はランダムに選択する．

EPを5台，許容ルール数差を20とし，上記2つのモデルでそれぞれ10回試行した．評価指標としてディスパッチャにおけるイベントコピー数を用いる．ラウンドロビン方式におけるコピー数に対する提案アルゴリズムにおけるコピー数の比が歩行者ITSモデルでは0.41，ランダムモデルでは0.69となり，ラウンドロビン方式と比較して提案方式が有効であることを確認した．イベントコピー数が少ないほどディスパッチャでのイベントコピーの負荷と送信負荷，EPでの受信負荷が低減される．歩行者ITSモデルでより有効であるのは，生成されるルールがいくつかの独立したグループに分けることができ，提案アルゴリズムがより効果的に働いたためである．

3.2. システムの性能評価

デバイスからのイベント増加，アプリケーションから設定されるルール数の増加に対して，システムにマシンを適宜追加することにより，品質を低下させずにサービスを提供できることを示す．

評価指標としてスループットと遅延時間を用いる．スループットは単位時間にシステムが処理したイベント数である．遅延時間はデバイスがイベントを生起してから，ルールマッチ通知がアプリケーションに到達するまでの時間である．

比較対象としてEP1台で処理する場合を用いる．EP1台の構成では，擬似デバイスと擬似アプリケーションが直接そのEPに接続する．比較対象のシステムを基本システムと呼ぶ．

評価では，デバイスとしてRFIDリーダを仮定し，RFIDから発生するイベントはタグID，リーダID，タイムスタンプから成るものとする．

3.2.1. システム構成と条件

提案方式のシステム構成では，ディスパッチャは1台か2台，EPは1台か3台か5台のいずれかとし，よってシステム構成は合計で6通りとなる．

基本システムと提案システムにおいて，擬似APと擬似デバイスにはスペックがPentium4 3.8GHz，1GB RAMの同一マシンを使用する．また，EP-CTL，ディスパッチャにはそれぞれPentium4 3.8GHz，1GB RAMのマシンを使用する．また，各EPにはPentiumM 1GHz，256MB RAMのマシンを使用する．

ルールモデルとして3.1節と同じ偏りのないランダムルールモデルを用いる．イベント生起，ルール設定

表 1 評価条件

設定項目		設定値
イベント生起	生起方法	ポアソン生起
	タグ ID の範囲	[0, 999]
	リーダ ID の範囲	[0, 99]
	ルール生起数	100, 1000
ルール設定	過去参照時間	60 秒
	タグ ID の範囲	[0, 999]
	リーダ ID の範囲	[0, 99]
	許容ルール数差	20

に関する条件を表 1に示す．イベント内，ルール内のタグ ID，リーダ ID は指定された範囲からランダムに決める．イベントのタイムスタンプはイベントが生起した時刻である．

3.2.2. 結果

ルール数が 1000，100 の場合の最大スループットをそれぞれ図 3，図 4に示す．最大スループットはイベント生起率5000イベント／秒から35000イベント／秒までの評価を実施した結果のうち，最大のものとする．

いずれのルール設定においても，提案方式の最大スループットは基本構成のものと比べて大きくなっており，提案方式の方がデバイスからのイベント増加にスケールすることが確認できた．ただし，その効果は設定するルールの種類，ルールの数によって異なる．

複合ルール数 1000 のときに，特に提案方式の効果が表れている．ディスパッチャが 1 台のときに，イベント処理サーバを 1 台，3 台，5 台と増加すると，最大スループットは，2313 イベント／秒，9991 イベント／秒，13162 イベント／秒と上昇しており，提案方式では，システムの構成を大規模化することによって，入力イベントにスケールできることが示せた．

一方，ルール数 100 の場合，ディスパッチャ台数を増加させることによる効果が出ている．これは，1 台のときにボトルネックとなっていたディスパッチャを 2 台にすることによりボトルネックが解消されたためである．ルール数 1000 の場合と比較して，ディスパッチャがボトルネックとなっているのは，ルール数が少ないためにデバイスからのイベントの大部分がディスパッチャで破棄され，EP での処理負荷に対してディスパッチャの処理負荷の比率が大きくなるためである．

このように，ルール数によって EP とディスパッチャにかかる負荷のバランスは異なり，その状況に応じて EP とディスパッチャに割り当てるリソースのバランスをとる必要がある．

遅延時間は 0.125 秒から 0.176 秒となっており，リアルタイム性が要求されるアプリケーションに利用できることが示された．ただし，遅延時間の算出には発散しない場合の計測値を用いている。

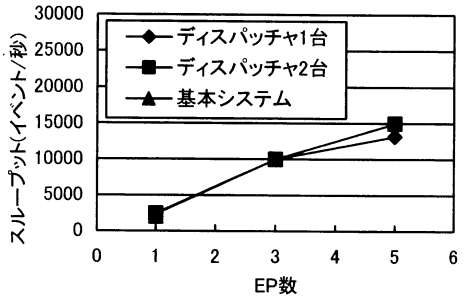

図 3 スループット（ルール数 1000）

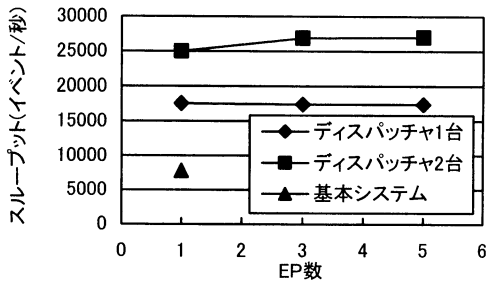

図 4 スループット（ルール数 100）

4. まとめ

本稿では，AP から設定されるルール数，デバイスから通知されるイベント数に対してスケーラビリティのあるコンテキストプラットフォームを提案した．ルール分配アルゴリズムが効率的であることを検証した．また，実機を用いた評価を行い，EP を 1 台から 5 台に増加させることによりスループットを 5.69 倍になることを確認し，リソースを追加することによりスケールすることを示した．

謝 辞

本研究は，総務省からの委託研究「ユビキタスネットワーク技術の研究開発」の成果である．

文 献

[1] H. Morikawa : “The design and implementation of context-aware services”, Proc. IEEE Int. Symposium on Application and the Internet Workshops, pp.293-298(2004)

[2] D. Luckham : “The Power of Events”, Pearson Education(2002)

[3] M. Mansouri-Samani, M. Sloman : “GEM: a generalized event monitoring language for distributed systems”, Distributed Systems Engineering, pp.96-108(1997)

[4] 渡部正文，伊東直子：“状況に基づいたプレゼンス通知ポリシ”，信学技報，NS2005-30，pp.25-28(2005)