Vol.2014-DPS-158 No.6
Vol.2014-CSEC-64 No.6
2014/3/6

コーディングレスな M2M ゲートウェイ上のセンサデータ集約を実現する Complex Sensor Data Aggregator の提案

森口昭[†1]　　中村雄一[†1]　山内利宏[†2]

M2M におけるセンサデータ収集では，通信量削減のため，M2M ゲートウェイからのセンサデータ送信前にヒストグラム加工等の集約処理を行うことが有効である．近年，センサデータ活用が進められた結果，収集データに対する分析結果や外部環境変化に応じて，出荷後の M2M ゲートウェイのデータ集約処理を変更する必要が生じている．しかし，集約処理の変更は M2M ゲートウェイのファームウェア更新で行う必要があり，コスト・リスク共に高い．これを解決するため，コーディングレスで M2M ゲートウェイ上の集約処理の定義・変更が可能な，センサデータ集約技術 Complex Sensor Data Aggregator（CSDA）を提案する．CSDA は，集約処理を入力・演算・出力の 3 プロセスに分解し，各プロセスの基本処理を予め搭載することにより，設定ファイル上で処理を選択するだけで集約処理を定義することが可能である．性能評価により，CSDA は M2M ゲートウェイを想定した環境上で，CPU 利用率 8 %，RAM 消費量 13%で動作可能であることを確認した．

1. はじめに

これまで，企業で利用されるデータは ERP や CRM 等の情報システムから得られる情報が中心だったが， 近年，M2M（Machine to Machine）技術の普及により，企業の製造・所有するマシン由来のデータも企業経営に活用されるようになった．例えば，製造業種では，自社製造機器の遠隔管理・故障予兆検知によるアフターサービスの拡充を目的として， 機器内の各種センサデータの収集・可視化・分析が行われている．具体的には，建設機械・農機メーカは製品の稼働状態管理システムを構築，故障前のエンジニア派遣や部品交換によるダウンタイム削減や位置情報監視による盗難防止機能の提供により，製品の付加価値と利益率向上を実現している[1][2][3].

これら M2M におけるセンサデータ収集は，M2M ゲートウェイがセンサからデータを取得し，インターネット上のサーバにアップロードすることによって行われる．ここで，M2M ゲートウェイとは，インターネットとセンサネットワーク間の異なるプロトコルを変換するセンサデータの中継機能やセンサネットワーク中の機器の起動・停止等の制御機能を担う装置である[4]．M2M ゲートウェイとサーバの通信には敷設の容易さから移動体通信回線や衛星通信が利用されることが多い[5][6]．M2M のトラフィックはコンシューマがスマートフォン等で利用する通信に比べ低く，その 1 回線あたりの ARPU は 300 円程度とされている[7]．近年，M2M 向けの低速かつ安価な通信サービスも登場しているが[8]，定額利用した場合は月額 800 円程度と実際の利用料金に比べ，いまだ高額であるため，従量課金契約が利用されることが多い．さらに，マシン 1 台毎に通信回線を敷設することから，回線数は膨大になるため，1 台あたりの通信量の削減は M2M サービスの提供者にとって必須となる．これに対し，機器メーカでは，M2M ゲートウェイ上でセンサデータを平均値やヒストグラム等に演算することにより，データ量を削減するセンサデータ集約処理が行われている [9][10][11].

近年，センサデータの活用が進められた結果，収集データに対する分析結果や外部環境変化に応じて，データ集約処理をダイナミックに変更可能なシステムが求められている[12]．例えば，センサデータを利用した機器故障検知では，故障検知に必要なデータを出荷時に特定することは困難なため，センサデータ集約処理の実装・搭載，収集，分析，集約処理の修正を繰り返すことにより，最適な予兆検知パターンを見出す PDCA サイクルが必要である．また，車両機械の場合，法改正により，急発進・急加速などの非効率な運転を検知するためのセンサデータ収集が追加で求められることもある[13].

しかし，集約処理を変更するためには，M2M ゲートウェイのファームウェアを遠隔更新する必要があり，プログラム修正・テスト及びファームウェア更新失敗時の復帰作業等，多大な工数とリスクを伴う．

これに対し，本研究ではコーディングレスでデータ集約処理の定義・変更が可能な，Complex Sensor Data Aggregator（CSDA）を提案する．CSDA は，センサデータ集約をデータの選別・加工閾値判定・連結送信の 3 プロセスに分解し，それぞれ設定ファイルで動作を定義可能とすることで，データ集約処理の変更を容易にするものである．本稿では，CSDA の設計方針・実装及び設定コストの評価と M2M ゲートウェイを想定した環境上での性能評価の結果を述べる．

2. センサデータ集約処理変更の課題

本章では，M2M ゲートウェイ上でのセンサデータ集約処理の概要を述べた後，ファームウェア更新による集約処理変更の課題を述べる．

2.1 センサデータ集約処理の概要

本節では，前処理となるセンサからのデータ取得処理と，

†1 (株)日立ソリューションズ
Hitachi Solutions, Ltd.
†2 岡山大学　大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

センサデータ集約処理について述べる．

2.1.1 センサからのデータ取得

M2M ゲートウェイとセンサはその用途により，様々なネットワークプロトコルで接続される．例えば，車両や医療機器等，通信のリアルタイム性が強く求められる機器内センサネットワークでは，優先制御機能を持つ CAN が利用される[14]．また，温度や湿度等の測定では，センサを広範囲に配置する必要があるため，デバイスが安価でノード間の無線通信中継機能を持つ Zigbee が利用される[15].

M2M ゲートウェイは，それらネットワークプロトコルを解釈するドライバを持ち，取得したセンサバイナリからセンサを識別するセンサ ID やセンサ値を含むバイナリを抽出した後に，データ集約処理を行う．例えば，CAN の場合は，SocketCAN[16]のようなドライバが受信した CAN データから送信元デバイスを表す CAN ID とセンサ値を含む CAN データバイナリを抽出し，データ集約処理にセンサ ID としての CAN ID とデータを渡す.

2.1.2 センサデータ集約処理

センサ ID とデータを入力として，M2M ゲートウェイ上でのデータ集約処理が行われる．集約処理は，(1) データ量を削減する加工処理，(2) 必要なデータのみを選別するフィルタリング処理，(3) サーバとの通信オーバーヘッドを削減するための複数データ連結処理，の 3 種類に分類できる．

(1) 加工処理

センサデータの値を平均値やヒストグラム等に加工することでデータを削減する．例えば，建設機械では，エンジン冷却水温や潤滑油温等の各センサデータを合計値，平均値・最大最小値及びヒストグラムに加工し，サーバに送信している [9]．同様に，農業機械においてもエンジン回転数センサや温度センサ等のデータを平均値・最大最小値に加工し，サーバに送信している[11].

加工処理は単一のセンサデータに対し，複数種類実行されることも考えられる．さらに，複数センサ値を組み合わせて加工する場合も考えられる．例えば，建設機械において，正常状態との乖離度を評価するため，特定の機器に取り付けられた複数の温度・圧力センサ毎に，分散値を演算後，それらの合計値を求めるといった処理が行われている[10].

(2) フィルタリング処理

フィルタリング処理は，センサ ID・センサデータ及び演算後のデータを利用して行われる．

・センサ ID によるフィルタリング

重要度の高いセンサに限定したデータ取得を目的として，特定のセンサ ID のセンサデータのみを送信対象とする．例えば，建設機械では，収集対象データを故障時の修復コストが大きいエンジンやトランスミッションのセンサに限定するといったフィルタリングを行っている[9].

・センサデータ・演算後のデータのフィルタリング

これらのフィルタリング処理の目的は，閾値判定により異常値のみを選択的に収集対象とすることである．例えば，建設機械においては，異常の兆候が見られる機器のセンサデータのみを収集するため，センサデータから演算により求めた正常値との乖離値が予め設定した正常範囲を逸脱した時のみセンサデータを送信するといった閾値判定を行っている[10].

(3) サーバ送信時の連結処理

サーバへデータを送信する際，通信プロトコルによるヘッダ情報が付与され，通信量が増大するため，サーバへの送信回数を減らす必要がある．そのため，収集対象のデータを複数種類連結して，データ識別のための ID を付与して送信する．

データを送信するタイミングについては，定期的送信，異常検知時のみの送信等，ユースケースによって異なる．建設機械の例では日毎に稼動情報を送信すると共に，エンジンのセンサ値の異常のような優先度が高いデータはイベント発生時に送信している[17].

2.2 ファームウェアによる集約処理変更の課題

以上で述べたデータ集約処理を変更するためには M2M ゲートウェイ上のファームウェア更新を行う必要がある．しかし，ファームウェア更新には，以下の課題がある．

(1) プログラム修正コスト

ファームウェア更新により集約処理を変更するためにはソースコードを修正・テストする工数が必要になる．産業向け IT の高度化に伴い，設置されるセンサ数は増加の一途を辿っており[18]，上述したセンサデータ集約処理の実装コスト及び修正コストもセンサ数に比例して増加し続けている．さらに，センサデータ集約処理に限らず，組込みソフト全体の開発費も肥大化傾向にあることから[19]，産業機械の本来機能ではないセンサデータ集約プログラムの修正にメーカが大きな工数を割くことは困難である．

(2) ネットワーク経由のファームウェア更新コスト

ファームウェア更新の手段としては，作業員が赴いての更新と，ネットワーク経由での更新が考えられる．作業員が赴いて更新する場合，人件費・出張費等のコストがかかる上に，機器の数が数千・数万となると，人手で行うことは非現実的であり，ネットワーク経由の更新が現実解となる．しかし，更新失敗時への対応と更新ファイル配信・永続化により，依然としてそのコストは大きい.

(a) 更新失敗リスクへの対応コスト

更新作業は専門家によって行われるとは限らないため，ネットワーク経由での更新には，更新中の電源断等によるファームウェア書き込み失敗により，ファームウェアが破壊されるリスクがある．よって，更新前のファームウェアの一時退避領域を用意し，更新失敗時には，技術者の操作により，更新前のファームウェアを復帰させなくてはなら

ない．ファームウェアの破壊は，同機器上で動作するセンサネットワークの制御機能も停止させるため，迅速な復帰作業が求められ，対応コストは大きい．

(b) 更新ファイル配信・永続化コスト

ファームウェア更新ではゲートウェイのシステム全体を更新することになるため，ファームウェアのファイルサイズが大きく，その配信にかかる通信コストは大きい．また，サイズが大きいため，配信中の通信途絶も発生しやすく，対策として，配信レジューム機能の搭載コストも必要である．また，永続化領域として最も利用される Flash ROM[20]には書き込み回数の制限があるため，サイズの大きいファームウェア更新を繰り返せば, Flash は急速に劣化する．そのため，劣化に備え，大容量の Flash を用意しなければならず，部品コストも大きくなる．

3. Complex Sensor Data Aggregator の提案

3.1 CSDA の設計方針

これまでのファームウェア更新では，プログラムの開発コストと遠隔更新にかかるコストが課題であった．これらの課題に対して，本章では，CSDA を提案する．

CSDA の構成を図 1 に示す．集約プロセッサと集約処理の内容を記した設定ファイルにより，センサデータ集約のコーディングレス化を実現し，更新モジュールによりネットワーク経由での更新を実現する．以下，これらの基本方針を記す．

3.1.1 集約プロセッサ

2.1 節で示した集約処理を網羅的にカバーするために，センサデータ集約処理シーケンスに着目し，入力処理・演算処理・出力処理の 3 プロセスに分解する．それぞれのプロセスに必要な基本処理は予め CSDA に搭載し，設定ファイルにて処理を選択するだけで動作を定義することを可能にする．

(1) 入力処理

入力処理では，プロトコルドライバから受け取ったセンサ ID とセンサ値を含むバイナリから，センサ ID を用いてフィルタリングを行い，必要なセンサ値を抽出する．

(2) 演算処理

入力処理で抽出されたセンサ値に対する処理を行う．平均・合計・最大・最小及びヒストグラム等の加工処理機能と，センサ値及び加工後の値に対するフィルタリング機能を持ち，それらを組み合わせた処理も実行する．

(3) 出力処理

出力処理では，演算処理後の複数データを連結して一定周期，もしくは異常検知時のイベント送信のため演算処理後即座にサーバに送信する．また，サーバ側でデータを識別するための ID も付与する．

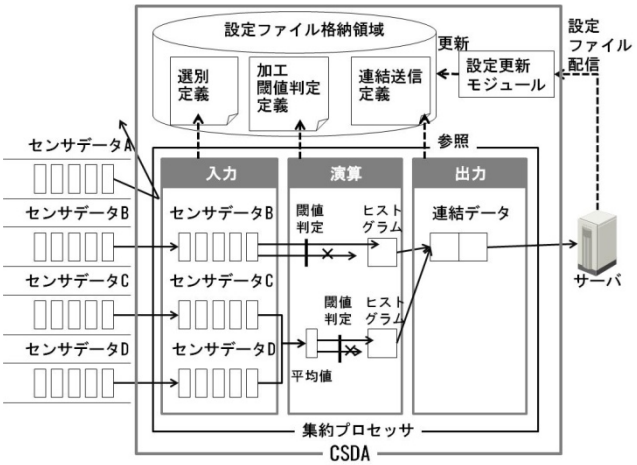

図 1 CSDA の構成

3.1.2 設定言語の設計

設定ファイルを記述する言語は，上記で述べた演算処理を表現可能なものでなければならない．

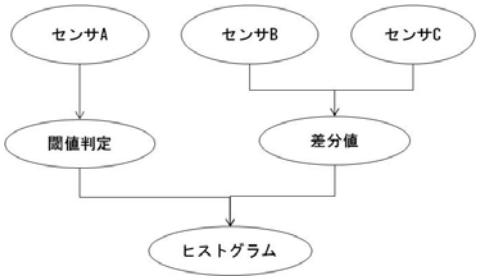

図 2 データ集約の処理フロー例

処理フローの例を図 2 に示す．この図は，センサ A のデータが異常値であった場合に，センサ B の値をセンサ C の値で差分補正した値からヒストグラムを作成するという演算処理を表している．このように，演算処理は複数のインプットデータから単一のアウトプットを生成する逆ツリー構造となるため，ツリー構造を表記可能な XML を設定ファイルの言語として採用する．

入力・演算・出力の 3 プロセスはそれぞれ選別定義・加工閾値判定定義・連結送信定義という設定ファイルに処理内容を記述する．

(1) 選別定義

取得対象となるセンサ ID と抽出するビット列の位置を定義する．

(2) 加工閾値判定定義

設定ファイルでは，入力処理後の複数データに対して，加工処理と閾値判定の任意の組み合わせを設定可能とする．さらに，閾値判定超過時に実行する出力処理を設定可能とする．

(3) 連結送信定義

連結する演算処理後のデータ，及び送信タイミングを記載する．

Vol.2014-DPS-158 No.6
Vol.2014-CSEC-64 No.6
2014/3/6

3.1.3 設定更新モジュール

ファームウェア更新の場合と同様に，設定のリモート更新についても設定更新失敗時のリスクへの対策が必要である．そのために，設定更新モジュールは CSDA のデータ集約処理と独立したプロセスで動作するようにし，受信処理後に永続化された設定ファイルを CSDA が起動時もしくは設定更新モジュールからの通知時に読み込むことにより変更が反映されるようにする．これにより，設定ファイルの更新失敗により，CSDA の動作不全に陥った場合もファイルの再配信と CSDA の再起動のみで復旧が可能である．

3.2 CSDA の実現方式

本節では，XML による選別定義・加工閾値判定定義・連結送信定義と，各処理を計算資源の限られた組込み機器である M2M ゲートウェイ上で動作させるための実現方式を述べる．

3.2.1 設定言語

選別定義・加工閾値判定定義・連結送信定義は，それぞれ definition タグの process 属性に Selection・Calculation・Sending を記載することで識別される．

プロセス間のデータの受け渡しは，各データに付与した ID を介して行う．設定ファイルでは，処理内容を記述する rule タグにて，その処理結果に ID を割り当てることができる．他の rule はこの ID を指定することにより，処理結果を参照することができる．

以下，各プロセスの設定言語の概要を示す．

(1) 選別定義

選別定義を図 3 に示す．入力処理では，センサネットワークから取得するセンサ ID と抽出対象となるビット列の位置を指定する必要がある．そこで，選別定義では，取得対象のセンサ ID とビット位置をそれぞれ rawId，offset，len（3 行目）で指定可能とする．ここで定義されていないセンサ ID のデータは不要データとしてフィルタリングされる．また，選別後データの ID は id（2 行目）で表され，続く加工閾値判定定義によるデータ取得時に指定される．

```
1  <definition process="Selection">
2      <rule id="50">
3          <extract rawId="10" offset="0" len="8"/>
4      </rule>
5  </definition>
```

図 3　選別定義

(2) 加工閾値判定定義

加工閾値判定定義を図 4 に示す．基本演算処理は，calc タグの method 属性（8 行目）にて平均値やヒストグラムといった演算名を指定し，arg タグ（9，10，11 行目）にてその実行パラメータを指定する．図 4 の指定では，最大値が 100，最小値が 0，級数が 20 のヒストグラムが生成される．なお，加工対象となるデータは seldata タグ（13，14 行目）に，選別定義で割り当てた ID を指定することで取り込むことができる．

複数加工処理の組み合わせは，基本演算処理を記述する calc タグを繰り返すことで記述でき（8，12 行目），また，複数センサ値の組み合わせは，seldata タグを複数指定することで表せる（13，14 行目）．閾値判定については，if タグ（3 行目）にて，比較条件（operator）と閾値（threshold）を記述，さらに比較対象の値を value タグ（4 行目），閾値判定条件に合致した場合の加工処理を action タグ(7 行目）で指定する．センサ値に対する閾値判定の場合は，value タグに直接 seldata タグを記述すればよい（5 行目）．また，演算結果に対する閾値判定の場合は，value タグ内に calc タグを記載する．図 4 の場合は，id : 50 の選別後センサデータ値が 100 以上であった場合，id : 51 と id : 52 のセンサデータ値の差分からヒストグラムを演算する処理が実行される．また，異常検知時のイベント送信のため，action タグ内には加工処理だけでなく，連結送信定義の ID も指定できる．この場合，calc タグではなく，send タグを記載し，id 属性で対象となる連結送信定義の ID 値を指定する．

将来，CSDA の標準搭載機能では実現できない演算処理が発生し，新規の演算処理を実装追加する可能性がある．これに備え，個々の演算処理名とそのパラメータ名は XML スキーマに含めずに，新規演算処理の追加をスキーマ変更無く行えるようにしている．例えば，パラメータ a を持つ新規演算処理 X が CSDA 内に追加された場合でも，method 属性に X，arg タグの key 属性に a を指定し，パラメータ値は value 属性に指定することで，設定が可能である．

```
1  <definition process="Calculation">
2      <rule id="100">
3          <if operator="greater" threshold="100">
4              <value>
5                  <seldata id="50"/>
6              </value>
7              <action>
8                  <calc method="histogram">
9                      <arg key="max" value="100"/>
10                     <arg key="min" value="0"/>
11                     <arg key="series" value="20"/>
12                     <calc method="subtraction">
13                         <seldata id="51" index="1"/>
14                         <seldata id="52" index="2"/>
15                     </calc>
16                 </calc>
17             </action>
18         </if>
19     </rule>
20 </definition>
```

図 4　加工閾値判定定義

(3) 連結送信定義

連結送信定義を図 5 に示す．出力処理では，連結対象となる演算結果データと送信データを識別する ID，送信タイミングを設定できる必要がある．連結する加工結果データは，cat タグ（3 行目）の caldata タグ（5，6 行目）にて，加工閾値判定定義で割り当てた id を記述する．ID は header タグ（4 行目）にて指定できる．また，送信タイミングに関しては send タグ内の method 属性(7 行目）と arg タグ(8，

9 行目）で指定する．周期送信する場合は，method に周期送信を表す periodic を記述し，arg タグで周期を記述する．図 5 の例では，100, 101 の ID を持つ加工データを連結し，ID：device001100 を付与し，30 分に一回送信する．閾値判定後のイベント送信処理の場合は，method にイベント送信をあらわす event を記述し，id 属性（2 行目）に加工閾値判定定義の send タグで記載する ID を指定する．

```
1   <definition process="Sending">
2       <rule id="1">
3           <cat>
4               <header data="device001"/>
5               <caldata id="100"/>
6               <caldata id="101"/>
7           </cat>
8           <send method="periodic"
                address="192.168.1.150" port="8081">
9               <arg key="unit" value="minute"/>
10              <arg key="interval" value="30"/>
11          </send>
12      </rule>
13  </definition>
```

図 5　連結送信定義

3.2.2　組込み機器向け実装

産業機械に搭載される M2M ゲートウェイのコストは製品価格に直結するため，コスト削減要求が厳しく，また高温多湿など過酷な環境で動作させるために，CPU や RAM に制限のある機器を利用しなければならない．計算資源の限られた組込み機器でも動作可能なように，CSDA では，センサデータを一時蓄積し，まとめて演算処理を行う演算処理バルク化と，設定ファイルのパース処理を軽量化するための設定ファイルのバイナリ化を行う．

(1)　演算処理バルク化

産業機械における主要なネットワークプロトコルである高速 CAN では，その最大通信速度は 1Mbps であり，1 データあたりのサイズは 64～128 bit である[21]. この場合，CSDA は約 100 us に 1 件のセンサデータを取得することになる．ところが，100 us 周期でセンサデータの演算処理を行えば，関数呼び出しや定義の参照による処理オーバーヘッドは甚大になる．本来，サーバに収集されたデータの使用目的は稼動監視や故障分析であり，機器に異常が見られた場合，部品交換等の対応は人手で行う必要があるため，センサデータの取得からサーバへの送信処理は秒単位の即時性があれば十分であると考えられる．よって，センサデータをメモリ上にバッファリングし，演算処理をバルクで行うことで，計算オーバーヘッドを削減する．なお，センサデータのメモリ上へのバッファリングにより，CSDA の RAM 使用量が利用可能な RAM 容量を超過することがないよう，センサデータを蓄積する領域サイズは機器毎に設定可能とする．

(2)　設定ファイルのバイナリ化

XML で記載された設定ファイルのパース処理はリソース消費が大きく，組込み機器で実行することは難しい．そこで，設定ファイルはサーバにてバイナリ圧縮した後に，CSDA に配信する方式を採る．バイナリ化された XML は前方からシーケンシャルに解釈可能なフォーマットをとることで，CSDA による設定ファイルのパース処理を軽量化する．例えば，図 4 に記載した加工閾値判定プロセスの設定ファイルの場合，処理順番は①3 行目の閾値判定，②12 行目の減算処理，③8 行目のヒストグラム生成処理となり，XML の記載順序と処理順序は一致しないが，これをサーバ上で処理順序に整列した上でバイナリ化することで，前方からの設定ファイルのパースを可能とする．

4.　CSDA の集約処理修正コスト及び性能評価

CSDA によるセンサデータ集約処理の修正コストをファームウェア更新と比較し評価する．さらに，M2M ゲートウェイ上に CSDA が搭載可能であることを性能評価によって確認する．

4.1　センサデータ集約処理変更のコスト

センサデータ集約処理変更に関わるコストを作業プロセス毎に比較する．データ集約処理更新の作業は（1）コード修正，（2）配信ファイル生成，（3）テスト機での動作確認，（4）リモート更新の 4 プロセスからなる．それぞれの比較結果を表 1 に示す．

表 1　CSDA とファームウェア更新のコスト比較

作業プロセス	CSDA	ファームウェア更新
(1)　コード修正	○	×
(2)　配信ファイル生成	－	－
(3)　テスト機での動作確認	○	×
(4)　リモート更新	○	×

以下，各プロセスにおけるコスト比較の詳細を述べる．

(1)　コード修正

新たにセンサ X のデータのヒストグラムを 1 種追加した場合のコード修正量を比較する．

ファームウェア更新では，以下の処理を C 言語にて実装する．

- センサ X の ID を持ったセンサデータを追加で取得し，対象ビット列を取り出し，ヒストグラムを生成するようコードに追加．さらに，既存の連結送信データにヒストグラムを追加で連結するように修正．

一方で，CSDA では以下のように定義を追記する．

① センサデータの追加取得は，図 3 の 2-4 行目で示したように，選別定義にセンサ X の ID とビット位置を記述することで行える．3 行の追記で修正できる．

② ヒストグラムの生成は図 4 の 8-16 行目で示したように method に histogram を指定し，ヒストグラムの最大値・最小値・級数をそれぞれ 1 行ずつ記述し，ヒスト

グラムに加える選別済みデータの ID を指定することで実行できる．6 行の追記で修正できる．

③ 生成したヒストグラムの送信は，図 5 の連結送信定義の 5 行目のように，加工閾値判定定義で設定した ID を既存の連結送信定義に 1 行追記すれば実行可能である．

CSDA では，新規センサデータのヒストグラムの生成とサーバへの収集を計 10 行の追記で設定できる．このように，CSDA では，選別・加工閾値判定・連結送信の 3 プロセスにおいて，処理名とそのパラメータを記載するだけで処理が追加可能なことから，C 言語実装に比べ，コーディングレスなデータ集約処理が可能であると考えられる．

(2) 配信ファイル生成

CSDA では，XML 定義ファイルをバイナリ変換するツールを提供する．ファームウェア更新では，C 言語コンパイルとファームウェアイメージ生成が必要である．いずれもコンソール上での１，２件のコマンド実行で完了でき，コスト差は無いと考えられる．

(3) テスト機での動作確認

配信ファイルはテスト機上での動作確認後，製品へ配信される．ファームウェア更新の動作確認の場合，ファームウェアイメージは数 MB にも及ぶため，テスト機への書き込みに時間がかかる．複数回の動作確認が必要な場合は，その時間はさらに大きくなる．CSDA の配信バイナリは，100 以上のセンサに対する設定ファイルであっても 10 KB 以下のサイズであることを確認しており，テスト機への書き込みに必要な時間はファームウェアに比べ，非常に小さい．そのため，CSDA の方が短時間で動作確認が可能である．

(4) リモート更新

2.2 節で示したように，ファームウェアのリモート更新には，(a) 失敗時のファームウェア復帰，(b) 通信途絶時のレジューム処理，(c) ファームウェアの通信，(d) Flash ROM に関わるコストが大きい．

一方で，CSDA における設定変更では，(a) 設定ファイル更新失敗しても，システム及び CSDA の設定ファイル受信は継続稼動可能なため，リモートからファイル再配信と CSDA 再起動で復旧が可能である．また，(3) で述べたようにファイルサイズが小さいため，(b) 更新失敗の発生確率も小さく，レジューム処理は不要であり，(c) 通信コストも小さく，(d) Flash ROM の劣化も抑制できる．

以上，(1) コード修正，(3) 動作確認，(4) 配信の 3 点で CSDA が優位にあることから，CSDA はファームウェア更新に比べ，低コストでセンサデータ集約処理の変更が可能である．

4.2 CSDA の M2M ゲートウェイ上での性能評価

4.2.1 評価環境

性能評価環境として，組込み開発ボード Armadillo-420 [22]を利用した．Armadillo-420 の仕様を表 2 に示す．M2M ゲートウェイの CPU クロックは 200～600 MHz，RAM は 1～64 MB であることが多く[23][24]，本ボードは実用環境と同等の性能を有する．

表 2　Armadillo-420 仕様

項目	仕様
CPU	Freescale i.MX25(ARM926EJ-S) 400 MHz
RAM	LPDDR SDRAM 64 MB
OS	Debian Lenny Linux カーネル 2.6

4.2.2 評価条件

本評価では，センサプロトコルに産業機械での主要プロトコルである高速 CAN を利用する．高負荷時でも動作することを確認するため，CAN の最大通信速度 1 Mbps でセンサデータを評価ボードに送信した際の CPU 使用率と RAM 消費量を評価した．以下，評価に用いたセンサデータの内容と，入力・演算・出力処理内容を説明する．

(1) センサデータ

車両に搭載されるセンサ数は最大で 150 程度であるため[25]，本評価では最も高負荷な，CAN を流れる全てのデータを集約対象とした場合を想定し，今後のセンサ数の増加を加味して 200 のセンサデータを集約した場合の性能測定を行う．

CAN データはヘッダサイズが 64 bit，データ格納領域のサイズが 0～64 bit である．データ格納領域には，CAN データを送信する車載機器が，自身に接続されている複数センサのデータを格納する．本評価では，単位時間当たりの集約処理数を増やすため，64 bit のデータ格納領域を全て利用し，16 bit 長の 4 種のセンサデータを格納する．

(2) 入力処理

入力処理では，50 種類の CAN データから 4 種ずつ合計 200 種類のセンサ値を取得し，3.2.2 節で述べたメモリ上のセンサデータのバッファに格納する．

(3) 演算処理

本評価の演算処理では，閾値判定処理と，標準加工処理の中でも高負荷なヒストグラム生成処理を組み合わせて実行する．具体的には，200 種のセンサ値の内，1 種のセンサ値は閾値判定に利用し，199 種のセンサ値は閾値判定結果を参照後，ヒストグラムを生成する．

(4) 出力処理

出力処理については，入力・演算処理によって集約されたデータに対する処理であるため，処理頻度は低く，時間単位，もしくは日単位である．よって，リソース消費に対

する影響も小さいため，評価から除外する．

4.2.3 評価結果

(1) CPU 使用率

3.2.2 節で示したように演算処理はバルクで行う．バルク化による処理オーバーヘッドの削減効果は 3.2.2 節で述べたメモリ上のセンサデータのバッファ量に依存するため，センサ当たりのバッファサイズを変化させながら測定を行った．測定結果を図 6 に示す．

バッファサイズが 32 Byte 以上の時，つまり 1 センサ辺り 16 個以上の値をバッファリングし，バルク処理を行った時，CPU 使用率は 4 %以下になる．M2M ゲートウェイの CPU クロックが最小の 200 MHz だった場合でも，評価ボードの CPU クロックは 400 MHz であることから，CPU 使用率はおよそ 8 %程度に収まると考えられる．

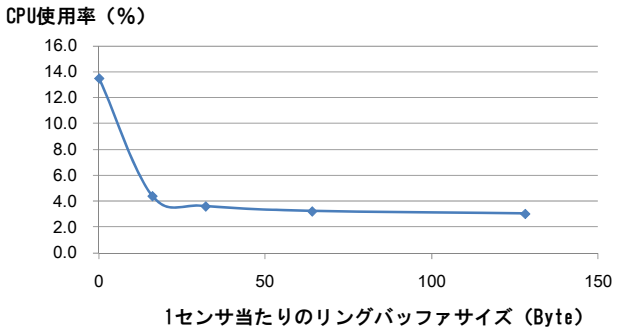

図 6　CSDA の CPU 使用率

(2) RAM 消費量

CPU 使用率の評価にて，バッファサイズ 32 Byte で十分な処理速度効率化が見られたため，同条件下で RAM 消費量を測定した．結果を図 7 に示す．

CSDA の RAM 消費量は全体で 128 KB である．図 7 中の定義ファイルと演算データは，メモリ上に展開中の定義ファイルと演算結果データの RAM 消費量を指す．演算データがセンサデータバッファに比較して大きくなっているのは，センサデータの演算方法が全て，処理結果サイズの大きいヒストグラム生成だからである．実際には平均・最大・最小等のサイズの小さい演算結果データも存在するため，CSDA の RAM 消費量は 128 KB 以下になると考えられる．M2M ゲートウェイの RAM 搭載量を最小の 1 MB と想定した場合，CSDA の RAM 消費は 13 %以下となる．

CSDA の M2M ゲートウェイ上の他のプログラムによる影響を，事例を元に考察する．建設機械では，位置情報から盗難を検知し，エンジンを停止させる制御を行っている[26]．また，自販機では，災害時にサーバからの指示により飲料を無料提供する機能を持つ[27]．いずれの場合も，実行頻度は低く，また，サーバからの受信データをネットワーク内の他の機器に転送するのみであり，CSDA への性能上の影響は軽微であると考えられる．よって，CSDA は M2M ゲートウェイのリソースでも十分に動作可能であると考えられる．

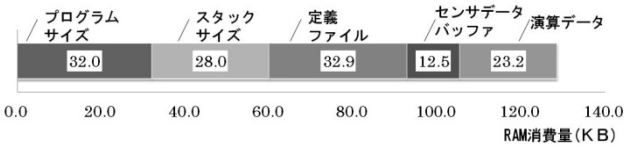

図 7　CSDA の RAM 消費量

5. 関連研究

SQL やスクリプトでセンサデータの集約処理を設定できる技術として，TinyDB と SensorWare がある[28][29]．TinyDB は，センサネットワークを仮想的に分散 DB に見立てて，SQL ライクな言語でセンサノードにデータ集約を行わせる技術である．また，SensorWare も同様に，センサノードからスクリプト言語でデータ集約を定義できる技術である． これらの技術は，センサノード側で集約処理を行わせ，通信量を削減することで，センサノードの電力消費量を削減できることが利点である．しかし，センサノード上の集約では，複数センサデータを組み合わせた演算処理を行うことができない．また，各センサノードに専用ミドルウェアをインストールする必要があるため，センサデバイスを構築済みのセンサネットワークに組み込むことは難しい．一方で，本技術は，複数センサを組み合わせた演算処理を定義可能であり，また，M2M ゲートウェイ上のソフトウェア改修だけでセンサデータ集約ロジックの変更が実現できるため，構築済みセンサネットワークへの導入も容易である．

6. おわりに

M2M ゲートウェイ上のセンサデータ集約処理変更のコストを削減することを目的とした，コーディングレスでデータ集約処理の定義・変更が可能な CSDA を提案した．

CSDA では，センサデータの集約処理シーケンスに着目し，集約処理を入力処理・演算処理・出力処理の 3 プロセスに分解する．それぞれのプロセスに必要な基本処理を CSDA に搭載し，設定ファイルにて処理を選択するだけで動作を定義することが可能である．これにより，M2M ゲートウェイ上のデータ集約処理の変更において，プログラム修正とネットワーク経由のファームウェア更新を必要としていた従来手法の課題を解決できる．

CSDA におけるセンサデータ集約処理変更に関わるコストを比較した結果，CSDA はファームウェア更新に比べ，低コストでセンサデータ集約処理を変更可能であることを示した．また，M2M ゲートウェイを想定した環境上で性能評価を行った結果，CSDA の CPU 利用率は約 8 %，RAM 消費量は最大 13%以下に抑えられており，M2M ゲートウ

ェイのリソースでも動作可能であることが確認できた．

今後の課題としては，C プログラムコードを用いたファームウェア更新との定量的なコスト比較や，本報告の評価条件よりも多くのセンサデータと演算処理を組み合わせた場合の性能評価が挙げられる．

参考文献

1) KOMATSU: KOMTRAX
http://www.komatsu-kenki.co.jp/service/product/komtrax/
2) 日立建機: Global e-Service
http://www.hitachi-kenki.co.jp/globaleservice/
3) YANMAR: SMART ASSIST
http://www.yanmar.co.jp/smartassist/
4) ETSI: Machine-to-Machine communications (M2M);Functional architecture
http://www.etsi.org/deliver/etsi_ts/102600_102699/102690/01.01.01_60
5) 和田恭: 米国における M2M の動向, IPA ニューヨーク便り (2012.3)
http://www.ipa.go.jp/files/000006081.pdf
6) 荒川修二: KOMTRAX STEP 2 の開発と展開, Komatsu Tech Rep, Vol.48, No.2, pp.8-14(2003)
http://www.komatsu.co.jp/CompanyInfo/profile/report/pdf/150-03_J.pdf
7) ATKearney：電波利用料制度に関する論点
http://www.soumu.go.jp/main_content/000217433.pdf
8) NTT Communications：Arcstar Universal One モバイル
http://www.ntt.com/vpn/mobile/data/m2m_fee.html
9) 村上卓, 西郷隆一, 大蔵泰則: 大型建設機械の健康管理システム（VHMS/WebCARE）の開発, Komatsu Tech Rep, Vol.48, No.2, pp.15-21(2003)
http://www.komatsu.co.jp/CompanyInfo/profile/report/pdf/150-04_J.pdf
10) 藤原淳輔, 鈴木英明: 建設機械の稼動データ収集装置, WIPO Patent, WO2013077309 A1 (2013.5.30)
11) 篠原吉彦, 坂本博文: 走行作業機械又は船舶の遠隔監視端末装置, WIPO Patent, WO2013080712 A1 (2013.6.6)
12) 野村総合研究所：2017 年 IT ロードマップ
http://www.nri.com/jp/news/2012/120529.html
13) 日立建機: 建設機械, 日立評論 (2014.1-2)
http://www.hitachihyoron.com/2014/01_02/index.html
14) CiA：Controller Area Network
http://www.can-cia.org/index.php?id=can
15) Zigbee Aliance：Interconnecting Zigbee & M2M Networks
http://docbox.etsi.org/workshop/2011/201110_m2mworkshop/03_m2mcooperation/zigbee_taylor.pdf
16) M. Kleine: SocketCAN – The official CAN API of the Linux Kernel Automotive Linux Summit Fall 2013, Edinburgh, UK (2013.10.24-25),
http://events.linuxfoundation.org/sites/events/files/slides/elce2013-Klein e-Budde.pdf
17) 瀧下芳彦, 村上勝彦, 関邦生, 森下一成: ICT が支える建設機械のライフサイクルサポート, 日立評論 Vol.94, No.5, pp.382-385(2012.5)
18) 富士キメラ総研：センサデバイスとソリューション市場調査
http://www.group.fuji-keizai.co.jp/press/pdf/120413_12036.pdf
19) 吉松則文, 村上和彰: 車載マイコンの現状について, 第 1 回 ISIT カーエレクトロニクス研究会, 福岡 (2008.11.28)
http://www.car-electronics.jp/files/2012/10/CurrentStateOfIn-vehicleMicrocomputer.pdf
20) Tech Village：組込み機器に欠かせなくなった不揮発メモリの最新動向
http://www.kumikomi.net/archives/2008/09/20nonvol.php
21) CYPRESS：コントローラ・エリア・ネットワーク 2.1
http://www.cypress.com/?docID=36794
22) アットマークテクノ：Armadillo-420
http://armadillo.atmark-techno.com/armadillo-420
日立超 LSI システムズ：産業機器向け通信モジュール
http://www.hitachi-ul.co.jp/system/cmodule/index.html
23) 田丸喜一郎: 組込みソフトウェア産業の現状と課題, 第 15 回組込みシステム開発技術展, 東京(2012.5.9-11)
24) http://www.ipa.go.jp/files/000004021.pdf
25) 日経 Automotive Technology：センサが決める車の性能
http://techon.nikkeibp.co.jp/article/TOPCOL/20091201/178189/
26) 宮内昭男: 災害復旧に対する IT を駆使した建機の活用事例, 建設機械, Vol.41, No.8, pp.13-16 (2005.8)
27) JR 東日本ウォータービジネス：次世代自販機
http://www.jre-water.com/pdf/100810jisedai-jihanki.pdf
28) S. Madden, M. Franklin, J. Hellerstein, W, Hong：TinyDB: An Acquisitional Query Processing System for Sensor Networks ：ACM Transactions on Database Systems, Vol.30, No.1(2005.3)
29) A. Boulis, C. Han, R. Shea, M. Srivastava：SensorWare:Programming sensor network beyond code update and querying：Pervasive and Mobile Computing, Vol.3, No.4, pp.386-412 (2007.8)