Vol.2016-IOT-32 No.20
2016/3/3

# 秘匿分散統計解析手法 *m*-cloud における分散トランザクション手法の設計と実装

中川 郁夫[1,2]　樋地 正浩[3]　菊池 豊[4]　福本 昌弘[4]　下條 真司[2]

概要：IoT (Internet of Things) 時代には数十億のセンサーやデバイスがインターネットに接続されるため, 機動性, 拡張性, 効率などを理由にクラウド上でデータを収集, 解析するケースが増えている. 一方でパブリッククラウドの利用にはデータ漏洩のリスクを懸念する声も多い. 我々は, クラウド上に秘匿分散しながらデータを分散収集することでプライバシ情報の漏洩リスクを低減しつつ非常に大規模な IoT データの収集と統計指標の計算を可能にする *m*-cloud 手法を提案してきた. 一方, *m*-cloud におけるデータ分割は障害や災害に伴う通信障害やデータ欠損に起因して無視できない誤差が発生しえる. 本稿では障害時のエラー回避手法として分散トランザクションの仕組みを提案し, そのような場合であっても *m*-cloud によって正確な統計指標を得ることを目指す.

キーワード：IoT, クラウドコンピューティング, 秘匿分散統計解析

# Design and implementation of distributed transaction for *m*-cloud - distributed privacy preserving statistical computation mechanism

Ikuo Nakagawa[1,2]　Masahiro Hiji[3]　Yutaka Kikuchi[4]　Masahiro Fukumoto[4]　Shinji Shimojo[2]

***Abstract:*** In the age of IoT, as known as Internet of Things, tens of billions sensors or devices will be connected to the Internet. For elasticity, scalability and efficiency, many service operators use cloud resources to collect data from IoT devices and analyze such data on the Internet. On the other hand, protecting privacy is an issue for using cloud resources. We proposed *m*-cloud, a distributed computation mechanism which enables us to collect data from IoT devices and calculate some statistical indices on public cloud environment while we reduce the risk of revealing privacy information. Although the mechanism is based on the idea of distributed processing on multiple independent cloud computers, we need additional features to avoid errors caused by communication failures or devices failures, in practical scenes. In this paper, we review the *m*-cloud computation and propose extensions for fault tolerant mechanism based on distributed transaction technique.

***Keywords:*** Internet of Things, Cloud Computing, Distributed Privacy Preserving Statistical Computation

## 1. はじめに

近年, IoT (Internet of Things) が注目を集めている. IoT によって, 2020 年には数百億ものセンサーやデバイスがネットに接続[*1] [*2] されるとの予測もある. また, 多数の IoT アプリケーションも登場してきた. 例えば家庭内の電力消費量の可視化, 室内温度のモニタリング, ヘルスケア機器での血圧計測など, あらゆる領域で IoT モデルによるサービスが考えられている.

我々は IoT アプリケーションにおけるデータの収集と解析手法に注目する. 特に, IoT によってパブリッククラウド

[1] インテック, Intec, Inc., Takaoka 933-8777, Japan
[2] 大阪大学, Osaka Univ.
[3] 東北大学, Tohoku Univ.
[4] 高知工科大学, Kochi Univ. of Technology

[*1] http://www.gartner.com/newsroom/id/2636073
[*2] http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

上の計算機資源をもちいてデータの収集と解析を行うケースにいて考える. パブリッククラウドは弾力性, 拡張性, 効率性に優れるなど多数のメリットを持ち, センサーの数が急速かつ指数的に増える可能性がある IoT アプリケーションの基盤に適している. 一方で, パブリッククラウドを利用する場合にはプライバシ情報の扱いが問題になるため, その漏洩対策は必須である.

我々は個々のセンサー情報を秘匿し, かつ複数のクラウド環境を用いて分散して統計解析を行うための仕組みである *m*-cloud [1] [2] を提案してきた. *m*-cloud は Secret Sharing Schemes[3] の応用である.

秘匿計算の分野では PPDM (Privacy Preserving Data Mining)[4] の研究も進んでいる. PPDM はオリジナルのセンサーデータを持つ複数のパーティ (party) が存在し, 互いに他のデータを知ることなく計算を行うことを目的とする. 一方 *m*-cloud はオリジナルのセンサーデータの集合がどこにも存在しない. ネット上に存在するのはセンサーデータの秘匿化された破片だけであり, 特定のシステムが攻撃されたとしても, オリジナルのデータ漏えいの心配がない.

匿名化 [5] は入力された情報量を減らすことによって安全性を高める. また, ランダマイズ技術 [5] は確率論によって近似的な統計指標を求めようとする. 一方 *m*-cloud は入力されたセンサーデータをもとに誤差のない正確な統計指標を計算することが可能であることを特徴とする.

既存の秘匿化技術とは異なり, *m*-cloud は膨大な数の入力データから実運用に耐えうる時間で統計指標を計算し, かつ, 急速な規模の拡大に耐えうるためスケールアウト型の拡張性を備えることを特徴とする. すなわち, *m*-cloud はクラウド時代の分散処理コンピューティングの手法を, 秘匿計算に応用しようとする試みである.

本稿では, *m*-cloud におけるエラー処理について述べる. 前述の通り, *m*-cloud では, アーキテクチャ上, 入力された膨大な数のセンサーデータに対して誤差のない正確な統計指標を計算することが可能である. 一方で, 実装・実運用を考慮した場合にはエラー処理を行うことが必須である. 著者らは DESTCloud [6] の研究において, 災害や障害時にネットワークやシステムへの影響をエミュレーションする仕組みの研究をしている. 同研究では災害・障害時の通信断などにインフラだけではカバーしきれないケースも議論しているが, このようなケースでは, 上位のプラットフォームやアプリケーションレベルでのエラー処理が重要になるとの指摘がある. 実際, *m*-cloud ではランダムシェアの考え方にもとづいてデータ分割をするため, 例えば, 通信障害やデバイスエラーなどに起因して分割後の破片が計算に反映されなかった場合など, 計算結果が予想外の誤差 (Error) を含んでしまう可能性がある.

本稿では, *m*-cloud における分散トランザクション手法について提案する. 分散トランザクションは, センサーから集められたデータの破片が正しくクラウド上に保存されたことを確認し, 統計指標の計算に誤差 (Error) が含まれないようにするための仕組みである. 前述の通り, 膨大な数のセンサーがさまざまな環境で接続される IoT アプリケーションにおいては, 本稿で提案する, 計算誤差を回避するための仕組みは, 実装上あるいは実用上, 極めて重要である.

本稿では, 2 節で前提条件及び本稿のゴールについて述べたのち 3 節で *m*-cloud 基本概念を再確認する. また、4 節で *m*-cloud のアーキテクチャとスケールアウト拡張について触れ, 5 節で分散トランザクションによって誤差のない統計指標の計算を実現する方法について提案する.

## 2. 前提とゴール

### 2.1 前提

本稿は以下のような IoT アプリケーションを想定する.

- サービス事業者は利用者のセンサーからデータを収集する. センサーデータには温度, 体重, 血圧など多様なものが想定される.
- サービス事業者は集めたデータについて統計指標を計算する. 主要な統計指標には, 合計, 平均, 分散, 偏差などを含む.
- サービス事業者はパブリッククラウド上にシステムを構築する. オンプレミスの環境は想定しない.
- 接続されるセンサーの数は数百万〜数億にもおよび, かつ急速に増える可能性がある.
- センサーから集められたデータにはプライバシ情報が含まれる可能性がある.

これらの前提条件は, 近年の IoT アプリケーションにとっては標準的であり, リーズナブルであると考えられる.

### 2.2 ゴール

*m*-cloud は秘匿分散統計解析手法として以下のような仕組みを提供することを目指している.

- センサーデータが持つプライバシ情報の漏洩リスクを低減しつつ, それらのデータに関する統計指標を正確に計算する.
- 上記の仕組みは, パブリッククラウド上の計算機資源を用いて実現する.
- 上記の仕組みは, センサーの数が急激に増えても対応ができるようスケールアウトの仕組みを有する.

さらに, 本稿ではセンサーに関わる通信障害やデバイス障害に起因する計算誤差を回避することを目指す.

- 通信障害, デバイス障害などの場合でも, 計算誤差のない, 正確な統計指標を計算することができる.

上記のゴールは IoT アプリケーションを実装し, 実際的な環境で運用する場合に極めて重要な意味を持つ.

## 3. Concept

第 2.2 節に記述したゴールを実現するため, 本稿では *m*-cloud および分散トランザクションの仕組みを提案する. ここでは, まず最初に *m*-cloud の中核である, 分散されたクラウド上で動作する統計処理の仕組みについて再確認する. 以下では, クラウド上の計算機資源であるクラウドコンピュータおよび複数のクラウドコンピュータを用いた秘匿分散統計解析の仕組みについて述べる.

### 3.1 クラウドコンピュータ

クラウドコンピュータはクラウド上にある計算機資源で, 多数のセンサーデータから統計指標を計算するために使用する. 図 1 にクラウドコンピュータの概念を図示する. Operator は IoT アプリケーションを提供するサービス事業者を意味する. インターネット上には多数のセンサーが接続しており, それらのセンサーがクラウドコンピュータにセンサーデータをアップロードする. クラウドコンピュータは受け取ったセンサーデータをもとに合計, 平均, 分散, 偏差などの統計指標を計算する. (なお IoT アプリケーションの詳細については本稿では触れない)

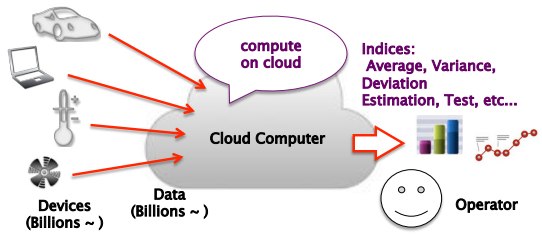


図 1 concept

クラウドコンピュータでセンサーデータを処理する場合, プライバシ情報の扱いが問題になる. 特に本モデルではサービス事業者とは異なる第三者が構築・運用するクラウドサービス上の計算機資源を用いてシステムを構成するため, データの漏洩への対策はさらに重要である. 前節の前提条件にも記載の通り, 集められたセンサーデータにはプライバシ情報を含んでいる可能性があるため, システムとしてプライバシ情報の漏洩への対策は必須である.

### 3.2 秘匿分散統計解析

プライバシ情報の漏洩リスクを抑えるため, *m*-cloud では秘匿分散統計解析の仕組みを導入する. 秘匿分散統計解析は Secret Sharing Schemes の応用の一つであり, センサーデータを秘匿化・分散して保持することでプライバシ情報の漏洩リスクを抑えつつ, 多数のセンサーデータから統計指標を計算することを可能にする.

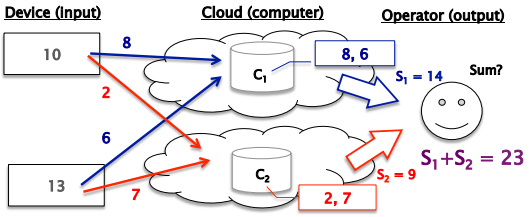


図 2 simple-example

図 2 に単純化した例を示す. 図では 2 つのセンサーデータを入力としており, それぞれのセンサーデータをランダムシェアの仕組みを使って複数 (図では 2 つ) のクラウドコンピュータにアップロードしている. 個々のクラウドコンピュータにアップロードされる値はランダム値としてのデータの破片でありプライバシに関わる情報は秘匿される. 一方で, 各クラウドコンピュータ上で入力値の合計を計算し, 最後に両方のクラウドコンピュータで計算した合計値を足しあわせることで, 個々のセンサーデータを知ることなく, 全体の統計指標 (例では合計, 平均が計算可能) を求めることが可能である.

上記の秘匿分散統計解析は, センサー数 $N$ が大きな値になっても同様に動作する. また, クラウドコンピュータの数を可変の $m$ とすることも可能である.

### 3.3 秘匿分散統計解析のメリット

秘匿分散統計解析にはいくつかのメリットがある. 以下, そのうちの主要な 3 点について述べる.

- 誤差のない正確な統計指標を求めることができる. 本手法は, 例えばオンライン選挙やオンラインでの試験などでも応用が可能である.
- プライバシ情報の漏洩リスクを低減する. 個々のクラウドコンピュータが扱うデータはランダムシェアによって分割されており, その情報量は $E(x_{ji}) = 0$ である.
- 実際的な計算量. クラウドコンピュータ上で実行するのは極めて単純な算術計算であり, 統計指標の計算に必要な計算機資源は極めて少なく, 実際的である.

## 4. *m*-cloud

*m*-cloud は膨大な数のセンサーデータを扱う IoT アプリケーションで秘匿分散統計解析を行う仕組みを提供する. 以下 *m*-cloud を用いてセンサー値の合計, 平均, 分散, 偏差などの基本的な統計指標を求めるための仕組みと, さらに DHT (Distributed Hash Table) を用いて, *m*-cloud をスケーラブルな拡張性を実現するための拡張について述べる.

### 4.1 *m*-cloud - 合計と平均

まず最初に *m*-cloud を用いてもっとも基本的な統計指標である合計 $s = \sum x_i$ および平均 $\bar{x}$ を計算する方法について述べる. ここでは $N$ 個のセンサーがインターネット経由で接続しており, 個々のセンサーがセンサーデータ $x_i$ $(i = 1, ..., N)$ をアップロードするものとする.

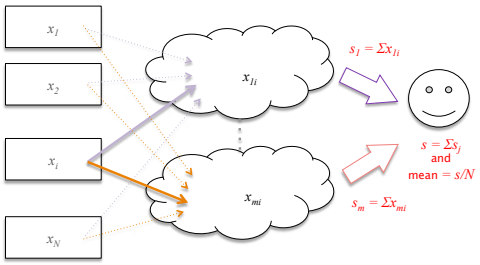


図 3　Summation and mean by m-cloud

図 3 に *m*-cloud の動作を示す. *m*-cloud では $m$ 個の独立なクラウドコンピュータ $C_1, ..., C_m$ が存在する. $i$ 番目のセンサーはそのセンサーデータ $x_i$ を $m$ 個のランダムな値 $x_{1i}, x_{2i}, ..., x_{mi}$ $(\sum_j x_{ji} = x_i)$ に分割し, $x_{ji}$ を $C_j$ に暗号化された通信路を用いてアップロードする.

$C_j$ $(j = 1, ..., m)$ は分割されたセンサーデータを受け取り, その合計を計算する. すなわち $C_j$ は $s_j = \sum_i x_{ji}$ を得る.

最後に $m$ 個のクラウドコンピュータが算出した合計値を合算し, $N$ 個のセンサーデータの合計値を得る.

$$\sum_j s_j = \sum_j \sum_i x_{ji} = \sum_i \sum_j x_{ji} = \sum_i x_i = s$$

また, 平均値 $\bar{x} = s/N$ も容易に求めることができる.

上記の手法において, 個々のクラウドコンピュータは, 単純に入力されたセンサーデータを合計しており, センサーデータがどのように分割されたか, 全体でいくつのクラウドコンピュータが存在するか, などについては一切関知しない. 個々のクラウドコンピュータの動作は極めて単純で簡単なものであるが, 入力されるデータはランダムシェアで分割された破片のみを扱うため, 個々のクラウドコンピュータが持つデータにプライバシ情報が含まれないことが情報理論的に裏付けられている.

### 4.2 *m*-cloud - 分散, 相関係数, など

*m*-cloud の仕組みを用いて分散を計算することも容易である. $i$ 番目のセンサーはセンサーデータの 2 乗値 $x_i^2$ を $m$ 個に分割する. 分割後の値 $x'_{1i}, ..., x'_{mi}$ $(\sum_j x'_{ji} = x_i^2)$ をそれぞれ $C_1, ..., C_m$ にアップロードすることで, 以下の通り $x_i^2$ の合計が計算できる.

$$\sum_j \sum_i x'_{ji} = \sum_i \sum_j x'_{ji} = \sum_i x_i^2 \qquad (1)$$

従って, 分散 $\delta^2$ 及び標準偏差 $\delta$ が求められる.

$$\delta^2 = \overline{x^2} - \bar{x}^2 = \frac{S}{N} - (\frac{s}{N})^2 \qquad (2)$$

なお, 簡単な拡張により, 複数の値 $x_i, y_i$ から相関係数を求めることも可能である.

### 4.3 DHT extension of *m*-cloud computation

前述の *m*-cloud の仕組みは固定的に $m$ 個のクラウドコンピュータを用いるよう設計されている. 一方, 膨大かつ急速に増えるセンサーからのセンサーデータのアップロードを処理するためには, 性能面での拡張性は必須である. *m*-cloud ではスケールアウト型の拡張性を実現するために

DHT (Distributed Hash Table)[7][8] 技術を応用する. DHT はハッシュテーブルの考え方を分散処理に応用したもので, 完全自立分散で, 中心的なサーバが存在しない処理モデルに従う. DHT は論理的なリング (輪) 構造を構成し, リングを構成する個々の処理ノードには担当するハッシュ値の範囲が割り当てられる. DHT は P2P (Point-To-Point) 技術の応用であり, 処理ノードを容易に追加・削除することでシステム全体の処理性能を動的に増強、縮退することができる. また, 処理ノードの障害時など, 動的に担当ハッシュ値の範囲を切り替えることにより, システム全体を停止させることなく障害対応できることを特徴とする. したがって, クラウドコンピュータの処理ノードに関する障害については DHT による冗長化、復旧の仕組みに従うものとし, 後述の議論では通信障害やデバイス障害など, ユーザデバイスに関連する障害に注目する.

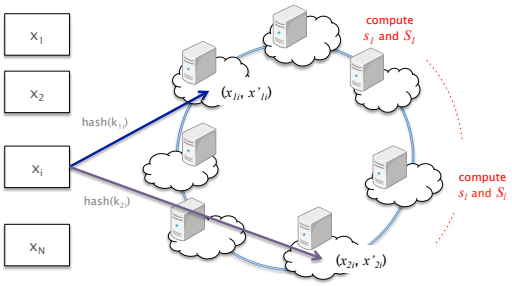


図 4　DHT extension

図 4 に *m*-cloud の DHT 拡張の概念を示す. 同図では, 統計解析処理を行うクラウドコンピュータが DHT リングを構成する処理ノードとして動作する. ここでは, $L$ 台のクラウドコンピュータ $C_1, ..., C_L$ によって DHT リングが構成されているとする. 個々のクラウドコンピュータは $K_l$ $(l = 1, ..., L)$ から始まるハッシュ値の範囲が割り当てられるものとする. すなわち, $C_l$ は $R_l = [K_l, K_{l+1})$ で表される範囲に含まれるキーについて処理の権限を持つ. なお, 簡単のため $K_{L+1} = K_1$ とする.

$i$ 番目のセンサーが, 適当に選んだ $m$ $(1 < m < L)$ をもちいて $x_i$ を $m$ 分割する. 分割後の値を $x_{1i}, ..., x_{mi}$ $(\sum_j x_{ji} = x_i)$ とする. 続いて, $m$ のキー $k_{1i}, ..., k_{mi}$ をランダムに生成し, 対

応するハッシュ値 $h_{1i}, ..., k_{mi}$ を計算する. 当該センサーは $m$ 個の値を対応するハッシュ値にもとづいてクラウドコンピュータにアップロードする. クラウドコンピュータ $C_l$ は $h_{ji} \in R_l = [K_l, K_{k+1})$ を満たすように選択される.

$N$ 個のセンサーがすべてのデータを対応するクラウドコンピュータにアップロードし終えた後, 個々のクラウドコンピュータ $C_l$ では入力されたデータの合計 $s_l$ を計算する. 最終的に, サービス事業者が $L$ 台のクラウドコンピュータで計算した合計値を合算し, 次の値を得る.

$$\sum_l s_l = \sum_i \sum_j x_{ji} = \sum_i x_i = s \qquad (3)$$

また $\bar{x} = s/N$ により $x_i$ の平均値を得る.

上記の例において, 入力を複数要素にすることも容易である. すなわち $i$ 番目のセンサーが $(x_i, x_i')$ を $m$ 分割してアップロードすることで, 合計 $s$ や平均 $\bar{x}$ に加えて、分散 $\delta^2$ や標準偏差 $\delta$ を求めることができる. 同様に, 複数センサーデータの相関係数なども計算可能である.

## 5. *m*-cloud における分散トランザクション

前述の通り, DHT 拡張によってクラウドコンピュータの処理ノードの冗長化や自動的な障害復旧が可能である. 一方, IoT アプリケーションでは対象となるセンサーの数は膨大であり, 多種多様な環境で接続されることからセンサーに関わる通信障害やデバイス障害などが相当数起こりえる. DESTCloud プロジェクト [6] でもさまざまな災害・障害のパターンを検討し, それらの災害・障害をエミュレーションする仕組みを研究しているが, 上位層で動作する IoT アプリケーションは, 独自に障害時の処理を規定することの必要性が指摘されている.

本節では, *m*-cloud において, 災害・障害にともなう通信障害やデバイス障害があった場合でも, 誤差のない正確な統計指標の計算を可能にするための分散トランザクションの仕組みについて提案する. 分散トランザクションは, 分割されたすべてのセンサーデータのアップロードをトランザクションとして認識することで, 欠損のあるデータを破棄し, 正しくアップロードできた値だけを統計指標に算入することを特徴とする. なお, 本稿で提案する分散トランザクションは DHT による分散処理技術を応用することにより, 個々のクラウドコンピュータが本来扱うべきセンサーデータのみを受け取り, プライバシ情報の秘匿性に影響をあたえることなくトランザクションを行うことも特徴である.

### 5.1 分散トランザクションの考え方

*m*-cloud における分散トランザクションは分散データベースにおける 2 層コミット [9] の考え方を参考にしている. ここでは $L$ 台のクラウドコンピュータが DHT リングを構成しているものとする. *m*-cloud で 2 層コミットを実現するため, まず, クラウドコンピュータにテーブルの概念を導入する. すなわち, センサーデータを扱う (アップロードする) ための *values* テーブル, および分散トランザクションのために使用する *flags* テーブルを定義する.

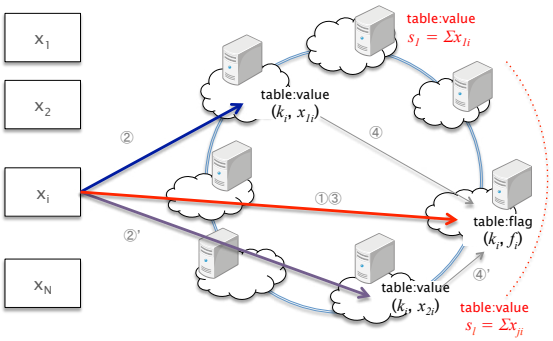


図 5 Basic idea of distributed transaction

図 5 に DHT *m*-cloud において分散トランザクションを行う場合の簡単な流れを示す.

$i$ 番目のセンサーは, (1) まず最初に *flags* テーブルにフラグ $f_i$ エントリを値 FALSE で作成する. $f_i$ は DHT のアルゴリズムに従いそのハッシュ値に対応するクラウドコンピュータ上に作成される. (2) その後 $m$ 個に分割したセンサーデータ $x_{1i}, ..., x_{mi}$ を対応するクラウドコンピュータの *values* テーブルにアップロードする. クラウドコンピュータは前述の DHT アルゴリズムに従って選択される. (3) 最後に $f_i$ フラグを TRUE にセットする.

なお, クラウドコンピュータでは *flags* テーブルにエントリが作成されたのち, あらかじめ定められた時間以上 TRUE にならなかったものについてはその値を強制的に不可逆な値 ERROR に置き換えるものとする.

クラウドコンピュータは *values* テーブルにアップロードされた $x_{ji}$ を合計値に算入する. 算入は, 対応するフラグ $f_i$ の値を対応するクラウドコンピュータの *flags* テーブルから取得して, フラグ値が TRUE であった場合に限って実行される. フラグ値が ERROR であった場合には処理を断念 (破棄), フラグ値が FALSE であった場合は一定時間後に再度処理を繰り返す.

上記の仕組みは, クラウドコンピュータ $C_l$ で計算されるセンセーデータの合計値 $s_l$ にあるセンサーの値 $x_{ji}$ が含まれる場合, 同じセンサーから得られた値 $x_{1i}, ..., x_{mi}$ すべてがクラウドコンピュータに正しくアップロードされたことを保証する. アップロードで欠損が発生した値は合計値には含まれず, 計算誤差がないことが保障される.

### 5.2 入力値の数

前述の分散トランザクションの仕組みを実装するためには, 合計値の計算に算入されたセンサーデータの数 $n$ を正しく把握する必要がある. これは $N$ センサーのうちいくつ

かのセンサーからのアップロードで欠損が発生し, 合計値の計算には反映されていない可能性があるためである.

合計値の算入に反映されたセンサーの数 $n$ は, 各センサーからアップロードする際に 1 を値としてもつ要素を加えることで計算できる. すなわち $(x_i, x'_i)$ の代わりに $(x_i, x'_i, 1)$ を $m$ 分割してアップロードすることで, 第 3 要素の合計は算入されたセンサーデータの数と等しくなる.

### 5.3 クラウドコンピュータ上の処理

分散トランザクション拡張の実現にはクラウドコンピュータの処理に工夫が必要である. 以下では *values* テーブル, *flags* テーブルそれぞれについてクラウドコンピュータが実装すべき機能について整理する. なお, 以下の処理はセンサーもしくはサービス事業者からの要求に応じてクラウドコンピュータ上で起動される処理を整理したものであり, いずれも, マルチスレッド環境において並列処理で実行されるものとする.

#### 5.3.1 *values* テーブル用の処理

**UPLOAD (from sensors):** センサーから分割されたセンサーデータを受け取る. 前述の通り, 複数要素のデータを受け取ることも可能とする. データにはフラグのキー $f_i$ が付与されているものとする. データ取得後, $f_i$ に対応するフラグの値を対応するクラウドコンピュータから取得, フラグの値が TRUE になるのを待って, 合算値にセンサーデータを参入する. なお, フラグの値が ERROR になった場合には, 当該センサーデータの処理を諦める.

**COMPUTE (from operator):** UPLOAD で受け取ったセンサーデータの処理がすべて終了するのを待って合計値を返す. 複数要素の入力があった場合には, 要素ごとの合計値を返す.

#### 5.3.2 *flags* テーブル用の処理と API

**CREATE (from sensors):** $f_i$ で特定されるフラグのエントリを作成する. 作成時, フラグの値は FALSE とする. 一定時間以内にフラグが TRUE にならなかった場合には, クラウドコンピュータによって不可逆な値 ERROR にセットされる.

**SET (from sensors):** $f_i$ で特定されるフラグのエントリの値を TRUE にセットする.

**GET (from cloud computers):** $f_i$ で特定されるフラグのエントリの値を取得する.

## 6. Conclusion

*m*-cloud はセンサーデータを秘匿分散することでプライバシ情報の漏洩リスクを低減し, かつパブリッククラウド上に分散して収集, 計算を行うことによって, 誤差のない正確な統計指標を計算することを可能にする. 本稿で提案した *m*-cloud の分散トランザクションは通信障害, デバイス障害などに起因する計算誤差を排除し, たとえ障害があったとしても正確な統計指標を計算することを可能にする.

IoT アプリケーションでは, 膨大な数のセンサーが多種多様な環境で接続され, 通信障害, デバイス障害などさまざまな異常事態がが起こりえる. *m*-cloud が想定する IoT アプリケーションにおいても本稿の仕組みによって統計解析における計算誤差を抑えることは極めて重要な意味を持つ. 今後は, 分散トランザクションのプロトタイプ実装と, 処理に与える影響の評価などを進めていく予定である.

## Acknowledgments

本研究にあたり有用なコメントをいただいた情報セキュリティ大学院大学の後藤教授, 湯淺教授, 及び T クラウド研究会の皆様に感謝いたします. また, 本研究の一部は科研 (No.26330085) 及び総務省による SCOPE(No.140201003) の支援をいただいたいます.

参考文献

[1] Nakagawa, I., Hiji, M., Kikuchi, Y., Fukumoto, M., Shimojo, S.: “m-cloud - Distributed statistical computation using multiple cloud computers,” IEEE COMPSAC Workshops, Pages 301-305, July, 2014.

[2] Nakagawa, I., Hiji, M., Kikuchi, Y., Fukumoto, M., Shimojo, S.: “DHT extension of m-cloud - scalable and distributed privacy preserving statistical computation on public cloud,” Proc. of IEEE COMPSAC, pp. 682-683, Jul., 2015.

[3] Bogdanov, Dan, Sven Laur, and Jan Willemson.: “Sharemind: A framework for fast privacy-preserving computations”, Computer Security-ESORICS 2008. Springer Berlin Heidelberg, pp 192-206. 2008.

[4] Agrawal, R. and Srikant, R.: “Privacy-Preserving Data Mining”, ACM SIGMOD Conference, Vol. 29, No. 2, pp. 439—450, 2000.

[5] Aggarwal, C., Yu, P.: “Privacy Preserving Data Mining: Models and Algorithm” Vol. 34, Advances in database systems, 2008.

[6] RICC/ITRC, “DESTCloud - Disaster Emulation Simulation Testbed for distributed systems such as Cloud computing environment”.

[7] Stoica, I., et al.: “Chord: A scalable peer-to-peer lookup service for internet applications”, ACM SIGCOMM Computer Communication Review Vol. 31, No. 4, 2001.

[8] Wang, L., et al.: “Measuring Large-Scale Distributed Systems: Case of BitTorrent Mainline DHT”, IEEE Peer-to-Peer. Sep., 2013.

[9] Yousef, J., Al-Houmaily, George Samaras: “Two-Phase Commit” Encyclopedia of Database Systems, pp.3204-3209, 2009.