データストリーム処理における近似的耐障害性保証のためのパーティショニング手法

高尾　大樹† 　杉浦　健人† 　石川　佳治† 　陸 　可鏡†

† 東海国立大学機構名古屋大学大学院情報学研究科　〒 464–8603 愛知県名古屋市千種区不老町

Email: {takao, lu}@db.is.i.nagoya-u.ac.jp, {sugiura, ishikawa}@i.nagoya-u.ac.jp

あらまし　エッジコンピューティング環境を想定したデータストリーム処理の耐障害性保証のアプローチとして，近似的な耐障害性保証に基づく手法がある．この手法は障害により損失したデータを後段の機器で近似的に復元することで，障害機器の復旧を待たずに処理を継続できるが，割当センサ数が増加したときの処理性能と復元精度の両立に課題がある．そこで本稿では，復元精度と計算コストのトレードオフを考慮したパーティション制御手法を提案する．本手法では近似的な耐障害性保証における誤差保証の枠組みを活用することで，ユーザ要求を満たして全てのセンサの処理結果を復元できる範囲で計算コストを最小化するようパーティション分割を動的に制御する．

キーワード　データストリーム処理，エッジコンピューティング，耐障害性，近似処理，並列処理．

1　はじめに

Raspberry Pi を始めとする安価なマイクロコンピュータの普及により，Internet of Things（IoT）およびそれらを活用するためのエッジコンピューティングに注目が集まっている．エッジコンピューティングではデータ通信量の削減や処理の負荷分散のため，集約やフィルタリングなどの簡単なデータ処理をネットワークエッジで行う [1]．例えばスマートシティでの環境センシングであれば，各区画や建物内に設置された環境センサが無線通信により送信した計測値を，各エッジが一定時間ごとに集約することで後段のアプリケーションとの通信量を削減できる．しかし大量の安価なマシンを使用するエッジコンピューティング環境（以下，エッジ環境と略す）ではその物理的な制約の強さから余剰リソースの集中的な管理が難しく，故障機器を人手で修理ないし交換しないと処理を継続できないという問題がある．エッジ環境において現実世界の変化に追従したサービスを継続して提供するためには，故障機器の復旧を待たずとも処理を継続できるような耐障害性保証のアプローチが求められる．

このエッジ環境における耐障害性の問題に対して，**近似的な耐障害性保証**に基づく手法 [2, 3] がある．この手法はセンサデータ処理において必ずしも誤差のない結果が必要ではない点に注目し，クラウド環境を想定した従来手法 [4–7] とは異なり近似処理を積極的に活用して障害に対処する．システム中のある機器に障害が発生したとき，時間的な相関およびセンサ間の相関を活用し，障害による損失データをその後段の機器で近似的に復元する．例えば図 1 においてエッジ 3 の障害によりセンサ X_E, X_F の処理結果を損失したとき，後段のゲートウェイで他の処理結果から近似的に復元する．故障機器の復旧に時間がかかるエッジ環境下において，障害機器の復旧を待つことなく高精度な近似処理結果を欠けることなくリアルタイムに提供できるため，クラウドなど後段のアプリケーションではデータクリーニングをすることなく直接分析処理などを行える．

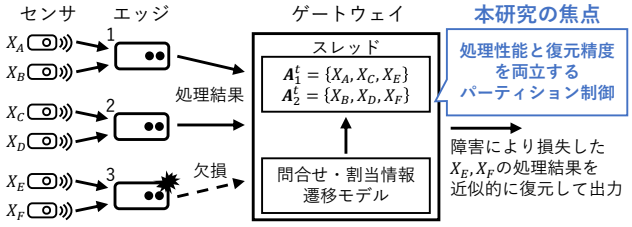

図 1　近似的な耐障害性保証の概要

しかしこの手法には処理性能と復元精度の両立に課題がある．物理的な制約により割当センサ数に上限があるエッジとは異なり，全エッジの出力を受け取るゲートウェイではシステム規模の拡大に伴い処理対象のデータ数が増加する．特に欠損データの復元に必要な行列演算のコストが大きいため，大量のデータを高速に処理するためには並列処理が必要になる．しかし，並列処理により各スレッドが復元に使用できるデータ量が減少することで，復元精度低下の恐れがある．以上より大規模環境への適用のためには，**全てのセンサの処理結果をユーザ要求を満たして復元できる範囲で計算コストを最小化**するようなパーティショニングが求められる．

そこで本稿では，復元精度と処理性能のトレードオフを考慮したパーティショニング手法を提案する．本手法は大きく**計算コストを最小化する初期割当の決定**，および**障害発生状況に応じた動的な割当制御**から構成される．パーティションを何度も大幅に変更すると，データの再送・再処理コストが大きくなり，処理性能が大きく低下してしまう．そこで本手法は，初期割当の段階から障害発生時の割当変更が最小限で済むよう調整するとともに，障害発生時に追加すべきセンサも高速に決定することで割当変更コストを最小限に抑えるよう工夫する．

本手法はヒューリスティックスを活用して，復元精度が高くなるようセンサの初期割当を決定する．環境センシングなどの

アプリケーションでは一般的に，物理的に近い位置のセンサほど同じエッジに割り当てられやすくかつ相関も強くなる傾向にあることに注目し，復元精度の近似的な評価関数を設計する．そしてこの関数に基づいてセンサを重複なく各スレッドに均等に配分することで，障害が発生していないときの計算コストを最小化するとともに，復元精度も高く維持できる．例えば図 1 では，スレッド 1 にセンサ X_A, X_C, X_E を，スレッド 2 にセンサ X_B, X_D, X_F をそれぞれ割り当てる．このとき任意エッジの障害に対して各スレッドは 1 つのセンサの結果を残りの 2 つのセンサの結果から復元できるため，1 つのセンサの結果のみから復元するよりも精度が向上する．よってユーザ指定の誤差要求を満たす確率も高くなるため，障害発生時の割当変更を最小限に抑えられるといえる．

また本手法は，近似的復元における誤差保証の枠組みを活用することで，障害状況に応じて効率的にセンサ割当を制御する．事前準備として，ユーザ要求の充足という観点から割当センサの優先度を計算し，各エッジに障害が発生したとき各スレッドへ最低限追加すべきセンサを求めておく．そして事前計算した情報に基づいて，各時刻の障害状況に対してユーザ要求を満たすために必要最小限のセンサを追加する．例えば図 1 においてエッジ 3 の障害によりセンサ X_E, X_F の処理結果を損失したとき，スレッド 2 はユーザ要求を満たして X_F の結果を復元できるが，スレッド 1 は X_E の結果を十分な精度で復元できないとする．このとき，事前に計算した情報を参考にスレッド 1 へセンサ X_D を追加で割り当てることで，センサ X_E の結果もユーザ要求を満たして復元する．

本研究の貢献は以下のとおりである．

- 損失データの復元処理を並列化する際の計算コストの評価関数を設計し，近似的な耐障害性保証における処理性能と復元精度を両立する理想的なパーティショニングを定式化する．
- センサの初期割当を決定するパーティショニングアルゴリズムを提案する．センサ間の相関および各エッジへのセンサ割当を考慮することで，復元精度と計算コストの両観点から適したセンサ割当を決定する．
- 障害状況に応じたセンサ割当制御手法を提案する．理論的な誤差保証の枠組みを活用し割当センサの優先度を事前計算することで，任意の障害状況に対してユーザ要求を満たすためのセンサ割当を効率的に決定する．
- 処理遅延と復元精度に注目して提案手法の性能を実験により評価する．

本稿の構成は以下の通りである．2 章では議論の準備として，エッジ環境を想定したデータストリーム処理の近似的な耐障害性保証について説明する．3 章では本研究における理想的なパーティショニングについて定式化する．そして 4 章では計算コストを最小化する初期割当のアルゴリズム，および障害状況に応じた動的な割当制御手法についてそれぞれ説明する．5 章では評価実験について述べ，6 章で関連研究について概説する．最後に，本研究のまとめと今後の課題を 7 章で述べる．

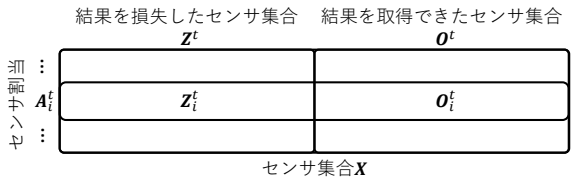

図 2　センサ集合を表す記号の関係性

2 準　　備

本章では議論の準備として，エッジ環境を想定した近似的な耐障害性保証に基づくデータストリーム処理システム [2,3] について説明する．本システムは，環境センシングのような物理的に固定されたセンサから無線通信により計測値を定期的に取得できる環境への適用を想定する．各エッジは割当センサから得られた計測値系列を時間窓で集約するなどの簡単なデータ処理により，後段のアプリケーションで処理しやすい形にクリーニングする．そして，ゲートウェイサーバや更に後段のクラウドサーバでは各エッジの処理結果に基づいて，データ分析やモニタリング，シミュレーションなどの様々なサービスを提供する．

本システムはエッジ環境の障害によるダウンタイムの長期化の課題に対して，処理結果にある程度の誤差を許容する代わりに，低遅延性と可用性を高い水準に保つ．時間的な相関およびセンサ間の相関を活用し，前段機器の障害による損失データをその後段の機器で近似的に復元することで，障害機器の復旧を待たずに高精度な近似処理結果を欠けることなくリアルタイムに提供する．先行研究では誤差保証の仕組みを考慮したウォータマーク制御 [2] や各エッジの近似処理による不確実性の考慮 [3] などの様々な性能改善の工夫がある．しかし本章では議論の単純化のため，エッジ障害のみを想定したゲートウェイでの近似的な復元処理に焦点を当てて，処理手順の詳細と理論的な誤差保証について説明する．

時刻 $t \in \mathbb{N}^+$ における n 台のセンサ $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$ の処理結果の真値を $\boldsymbol{x^t} = \langle x_1^t, x_2^t, \ldots, x_n^t \rangle$ と表す．ゲートウェイでは，時刻 t において障害の発生しているエッジに割り当てられたセンサ $\boldsymbol{Z^t} \subseteq \boldsymbol{X}$ の処理結果は全て損失し，センサ $\boldsymbol{O^t} = \boldsymbol{X} \setminus \boldsymbol{Z^t}$ の処理結果として $\boldsymbol{o^t}$ が得られるとする．ここで時刻 t におけるスレッド i のセンサ割当を $\boldsymbol{A_i^t} \subseteq \boldsymbol{X}$ とするとき，スレッド i では $\boldsymbol{O_i^t} = \boldsymbol{O^t} \cap \boldsymbol{A_i^t}$ の処理結果から $\boldsymbol{Z_i^t} = \boldsymbol{Z^t} \cap \boldsymbol{A_i^t}$ の処理結果を近似的に復元する．これらセンサ集合を表す記号の関係性を図 2 に整理する．例えば図 1 ではエッジ 3 の故障により $\boldsymbol{Z^t} = \{X_E, X_F\}$ の処理結果を損失するため，$\boldsymbol{O^t} = \{X_A, X_B, X_C, X_D\}$ の処理結果のみが得られる．このとき $\boldsymbol{A_1^t} = \{X_A, X_C, X_E\}$ のセンサが割り当てられているスレッド 1 では，$\boldsymbol{O_1^t} = \{X_A, X_C\}$ の処理結果から $\boldsymbol{Z_1^t} = \{X_E\}$ の処理結果を近似的に復元する．なお，本章では各スレッドへのセンサ割当は既に与えられているものとし，具体的なパーティショニングアルゴリズムは 4 章で述べる．

エッジ障害によるデータ損失の問題に対して，センサ間に存在する相関関係および時間的な相関を活用し，得られた一部の処理結果から欠損値を確率分布の形で復元する．本稿では，セ

ンサ $\boldsymbol{X}$ 間の相関関係を**多変量正規分布** $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ によりモデル化する．ここで，$\boldsymbol{\mu}$ と Σ はそれぞれ全センサの処理結果の期待値ベクトルと分散共分散行列を表す．また時間的な相関に関する議論の単純化のため，本稿ではマルコフ性を仮定する．すなわち，時刻 $t+1$ の各センサの処理結果 $\boldsymbol{X}^{t+1}$ は時刻 t の処理結果 $\boldsymbol{X}^t$ のみに依存し，その相関関係は遷移モデル $p(\boldsymbol{X}^{t+1} \mid \boldsymbol{X}^t)$ として表現できるとする．

以上の想定のもと，ゲートウェイでは次の手順を反復的に繰り返すことで各センサの処理結果を近似的に計算する．

（1） 時刻 t で受け取った一部のセンサの処理結果を用いて，事前確率分布から全センサの処理結果の事後確率分布を計算する．

（2） 遷移モデルおよび時刻 t の事後確率分布から次の時刻 $t+1$ における事前確率分布を計算する．

以降では各処理の具体的な計算方法について述べる．

時刻 t での事前確率分布として正規分布 $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ が与えられたとき，時刻 t でスレッド i が受け取ったセンサ $\boldsymbol{O}_i^t$ の処理結果 $\boldsymbol{o}_i^t$ からセンサ $\boldsymbol{Z}_i^t$ の事後確率を表す正規分布 $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Z}_i^t|\boldsymbol{o}_i^t}, \Sigma_{\boldsymbol{Z}_i^t|\boldsymbol{o}_i^t})$ は次のとおり求められる [8].

$$\boldsymbol{\mu}_{\boldsymbol{Z}_i^t|\boldsymbol{o}_i^t} = \boldsymbol{\mu}_{\boldsymbol{Z}_i^t} + \Sigma_{\boldsymbol{Z}_i^t\boldsymbol{O}_i^t}\Sigma_{\boldsymbol{O}_i^t\boldsymbol{O}_i^t}^{-1}\left(\boldsymbol{o}_i^t - \boldsymbol{\mu}_{\boldsymbol{O}_i^t}\right) \tag{1}$$

$$\Sigma_{\boldsymbol{Z}_i^t|\boldsymbol{o}_i^t} = \Sigma_{\boldsymbol{Z}_i^t\boldsymbol{Z}_i^t} - \Sigma_{\boldsymbol{Z}_i^t\boldsymbol{O}_i^t}\Sigma_{\boldsymbol{O}_i^t\boldsymbol{O}_i^t}^{-1}\Sigma_{\boldsymbol{O}_i^t\boldsymbol{Z}_i^t} \tag{2}$$

ここで，各記号の添字はベクトルないし行列から添字に対応する次元を抽出したことを示す．例えば，$\boldsymbol{\mu}_{\boldsymbol{Z}_i^t}$ は $\boldsymbol{Z}_i^t$ に対応する次元の値を期待値ベクトル $\boldsymbol{\mu}$ から抽出したもの（$\boldsymbol{Z}_i^t$ の期待値ベクトル）であり，$\Sigma_{\boldsymbol{Z}_i^t\boldsymbol{O}_i^t}$ は分散共分散行列 Σ の $\boldsymbol{Z}_i^t$ に対応する行から $\boldsymbol{O}_i^t$ に対応する列を抽出したもの（$\boldsymbol{Z}_i^t$ と $\boldsymbol{O}_i^t$ の共分散を表すブロック行列）である．また，時刻 t における $\boldsymbol{A}_i^t$ の事後確率分布として $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{A}_i^t|\boldsymbol{o}_i^t}, \Sigma_{\boldsymbol{A}_i^t|\boldsymbol{o}_i^t})$ が得られる．ただし，センサ $\boldsymbol{O}_i^t$ に対応する期待値は $\boldsymbol{o}_i^t$ であり，分散は全て 0 となる．

時刻 $t+1$ における事前確率分布は，遷移モデルおよび時刻 t での事後確率分布を用いて次のとおり求める．

$$\begin{aligned} &p(\boldsymbol{A}_i^{t+1} \mid \boldsymbol{o}_i^1, \cdots, \boldsymbol{o}_i^t) \\ &= \int p(\boldsymbol{A}_i^{t+1} \mid \boldsymbol{A}_i^t = \boldsymbol{x}) p(\boldsymbol{A}_i^t = \boldsymbol{x} \mid \boldsymbol{o}_i^1, \cdots, \boldsymbol{o}_i^t) d\boldsymbol{x} \end{aligned} \tag{3}$$

特に時刻 $t+1$ および t の処理結果の同時確率分布 $p(\boldsymbol{X}^{t+1}, \boldsymbol{X}^t)$ が $\mathcal{N}(\boldsymbol{\mu}'', \Sigma'')$ として表せるとき，式 (3) を計算することで時刻 $t+1$ の事前確率として次の正規分布 $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{A}_i^{t+1}|\boldsymbol{o}_i^t}, \Sigma_{\boldsymbol{A}_i^{t+1}|\boldsymbol{o}_i^t})$ が得られる [3].

$$\boldsymbol{\mu}_{\boldsymbol{A}_i^{t+1}|\boldsymbol{o}_i^t} = \boldsymbol{\mu}_A'' + \Sigma_{AB}''\left(\Sigma_{BB}''\right)^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{A}_i^t|\boldsymbol{o}_i^t} - \boldsymbol{\mu}_A''\right) \tag{4}$$

$$\begin{aligned} \Sigma_{\boldsymbol{A}_i^{t+1}|\boldsymbol{o}_i^t} =& \Sigma_{AA}'' - \Sigma_{AB}''\left(\Sigma_{BB}''\right)^{-1}\Sigma_{BA}'' \\ &+ \Sigma_{AB}''\left(\Sigma_{BB}''\right)^{-1}\Sigma_{\boldsymbol{A}_i^t|\boldsymbol{o}_i^t}\left(\Sigma_{BB}''\right)^{-1}\Sigma_{BA}'' \end{aligned} \tag{5}$$

ただし，$\boldsymbol{\mu}''$ および Σ'' の添字について A は $\boldsymbol{A}_i^{t+1}$ に対応する次元を，B は $\boldsymbol{A}_i^t$ に対応する次元をそれぞれ抽出することを示す．

処理結果を確率分布として計算することで，誤差上限を理論的に保証できる．センサ $X \in \boldsymbol{X}$ の処理結果が μ_X である確率はある誤差上限 e を用いて以下のように求められる．

$$P\left(X \in [\mu_X - e, \mu_X + e]\right) = \int_{\mu_X - e}^{\mu_X + e} p\left(X = x\right) dx \tag{6}$$

ここで，$p(X)$ は正規分布 $\mathcal{N}(\mu_X, \Sigma_X)$ に対する確率密度関数を表す．式 (6) より，ユーザ要求として要求信頼度 δ が与えられたとき，$P(X \in [\mu_X - e', \mu_X + e']) = \delta$ を解くことで，要求信頼度 δ を満たす最小の復元誤差 e' を計算できる．このとき，δ の確率で X の真値が $\mu_X \pm e'$ 以内に存在することを保証する．ただし，$\Sigma_X = 0$ のときは処理結果の真値が得られているため，常に $e' = 0$ であり復元結果に誤差は含まれない．

3 問題定義

本章では，2 章で述べた近似的な耐障害性保証に基づくデータストリーム処理システムにおける理想的なパーティショニングを定式化する．式 (5) などの欠損データの復元に必要な行列演算のコストが大きいため並列処理により処理性能を向上させたいが，各スレッドで復元に使用できるデータ量減少による復元精度低下の恐れがある．この処理性能と復元精度のトレードオフに対して本稿では，全てのセンサの処理結果をユーザ要求を満たして復元できる範囲で計算コストを最小化するセンサ割当を理想的なパーティショニングとみなす．

理想的なパーティショニングを定式化するため，はじめにユーザ要求の充足条件を整理する．ただし，全 m 個のエッジに対するセンサ割当 $\mathcal{X} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_m\}$，および時刻 t の事前確率分布の分散共分散行列 Σ は既知であるとする．時刻 t のエッジ障害によりセンサ $\boldsymbol{Z}^t \subseteq \boldsymbol{X}$ の処理結果を損失するとき，スレッド i ではセンサ $X \in \boldsymbol{A}_i^t \cap \boldsymbol{Z}^t$ の復元にセンサ $\boldsymbol{O}_i^t = \boldsymbol{A}_i^t \setminus \boldsymbol{Z}^t$ が使用でき，次式のとおり復元結果の分散 $\sigma_X(\boldsymbol{A}_i^t)$ を求められる．

$$\sigma_X(\boldsymbol{A}_i^t) = \Sigma_{XX} - \Sigma_{X\boldsymbol{O}_i^t}\Sigma_{\boldsymbol{O}_i^t\boldsymbol{O}_i^t}^{-1}\Sigma_{\boldsymbol{O}_i^tX} \tag{7}$$

ここで $\sigma_X(\boldsymbol{A}_i^t)$ は実際に得られる処理結果の値に依存しないため，復元で使用するセンサが分かれば事前計算できる点に注意する．また 2 章で述べたとおり，ユーザ要求として与えられた要求信頼度 δ および許容誤差 ϵ を満たすために必要な分散の閾値 σ_θ も事前計算できる．以上より，任意のセンサ X に対して $\sigma_X(\boldsymbol{A}_i^t) \leqq \sigma_\theta$ となるようなセンサ割当を決定できれば，ユーザ要求を満たして損失データを復元できるといえる．[1]

次に各スレッドの計算コストの評価関数を設計する．スレッド i は 2 章で述べたとおり，式 (1), (2) の損失データの復元と式 (4), (5) の遷移モデルの更新を毎時刻繰り返す．ここで時刻 t の割当センサ数を $|\boldsymbol{A}_i^t|$ と表すとき，データを受け取ったセンサ数 $|\boldsymbol{O}_i^t|$ および復元対象のセンサ数 $|\boldsymbol{Z}_i^t|$ はそれぞれ $|\boldsymbol{A}_i^t|$ 以下であるため，$|\boldsymbol{A}_i^t| \times |\boldsymbol{A}_i^t|$ 行列の計算コストでスレッド i での最悪計算量を見積もれる．$|\boldsymbol{A}_i^t| \times |\boldsymbol{A}_i^t|$ 行列の A, B に対して逆行列 A^{-1} と行列積 AB の計算には共に $2|\boldsymbol{A}_i^t|^3$ ステップ必要であ

1：ユーザ要求を満たせない場合のシステムの挙動として，満たせないまま近似的復元を続けるか処理自体を止めるかを決める必要はあるが，本稿では議論の対象外とする．

る [9] ことから，スレッド i の計算量を $O(|\boldsymbol{A}_i^t|^3)$ と見積る．以上より，計算コストの評価関数 f_{cost} を次のとおり定義する．

$$f_{cost}\left(\boldsymbol{A}_i^t\right) = |\boldsymbol{A}_i^t|^3 \tag{8}$$

以上を踏まえて本システムにおける時刻 t の理想的なパーティショニングを，次の最適化問題を解くことで得られるセンサ割当 $\mathcal{A}^t = \{\boldsymbol{A}_1^t, \boldsymbol{A}_2^t, \ldots, \boldsymbol{A}_m^t\}$ と定義する．

$$\begin{aligned} \underset{\mathcal{A}^t}{\text{minimize}} \quad & \sum_{\boldsymbol{A}_i^t \in \mathcal{A}^t} f_{cost}\left(\boldsymbol{A}_i^t\right) \\ \text{subject to} \quad & \forall \boldsymbol{A}_i^t \in \mathcal{A}^t, \forall X \in \boldsymbol{A}_i^t, \sigma_X(\boldsymbol{A}_i^t) \leqq \sigma_\theta \\ & \forall X \in \boldsymbol{X}, \exists \boldsymbol{A}_i^t \in \mathcal{A}^t, X \in \boldsymbol{A}_i^t \end{aligned} \tag{9}$$

つまり各時刻において，各センサを少なくとも 1 つのスレッドに割り当て，かつユーザの誤差要求を満たす範囲で計算コストを最小化するようなパーティショニングを求めたい．

4 提案手法

上述の最適化問題の厳密解を毎時刻求めるのは計算コストの観点から現実的でないため，本章では処理性能と復元精度の両立を目指した効率的なパーティショニング手法を提案する．現実のアプリケーションを想定したときエッジ障害が発生しない時間の方が長いと考えられるため，センサの初期割当を決める段階では通常時の計算コスト最小化を優先してセンサを重複なく各スレッドに均等に配分する．このとき単にセンサをランダムに割り当てるのではなく，今後障害が発生した際にセンサの追加を最小限に抑えられるよう初期配置の段階から割当を工夫する．そして各時刻の障害状況に応じて必要最小限のセンサを追加で割り当てることで，可能な限りユーザ要求を満たして損失データを復元する．以降では計算コストを最小化する初期割当の決定，および障害発生状況に応じた動的な割当制御の詳細をそれぞれ説明する．

4.1 計算コストを最小化する初期割当の決定

まずはじめにセンサ割当の指針として，単一エッジ障害を想定した復元精度に基づく評価関数を設計する．エッジ j の障害により，スレッド i ではエッジ j に割り当てられているセンサ $\boldsymbol{X}_j \in \mathcal{X}$ の処理結果を損失する．このとき，センサ $X \in \boldsymbol{A}_i^0 \cap \boldsymbol{X}_j$ の復元にはセンサ $\boldsymbol{O}_{ij} = \boldsymbol{A}_i^0 \setminus \boldsymbol{X}_j$ が使用でき，復元結果の分散 σ_X は次式のとおり求められる．

$$\sigma_X = \Sigma_{XX} - \Sigma_{X\boldsymbol{O}_{ij}} \Sigma^{-1}_{\boldsymbol{O}_{ij}\boldsymbol{O}_{ij}} \Sigma_{\boldsymbol{O}_{ij}X} \tag{10}$$

ここで σ_X およびユーザ要求充足に必要な分散の閾値 σ_θ の差分 $\Delta\sigma_X = \sigma_\theta - \sigma_X$ が非負値となるようなセンサ X は単一エッジ障害時にもユーザ要求を充足できると分かる．理想的には割当センサ全てにおいてユーザ要求を充足したいため，スレッド i の復元精度の評価関数 f_{acc} を差分の最小値と定義する．

$$f_{acc}\left(\boldsymbol{A}_i^0 \mid \mathcal{X}, \Sigma\right) = \min_{X \in \boldsymbol{A}_i^0} \Delta\sigma_X \tag{11}$$

Input: センサ集合 $\boldsymbol{X}$，各エッジへのセンサ割当 $\mathcal{X}$，
モデルの分散共分散行列 Σ，スレッド数 N
Output: 各スレッドへのセンサ初期割当 $\mathcal{A}^0$

```
/* 起点とするセンサを各スレッドへ割り当て                    */
1 for i = 1 to N do
2     X_max ← arg max_{X∈X} Σ_XX
3     A_i^0 ← {X_max}
4     X ← X \ {X_max}
/* 評価関数に基づいて残りのセンサを各スレッドへ割り当て      */
5 while X ≠ φ do
6     for i = 1 to N do
7         X_max ← arg max_{X∈X} f_acc(A_i^0 ∪ {X} | 𝒳, Σ)
8         A_i^0 ← A_i^0 ∪ {X_max}
9         X ← X \ {X_max}
```

図 3　センサの初期割当を決定するアルゴリズム

上述の評価関数に基づく貪欲的なパーティショニングアルゴリズムを図 3 に示す．本アルゴリズムは大きく 2 つのステップから構成される．最初のステップでは，各スレッドに対して起点となるセンサを 1 つずつ割り当てる（1–4 行目）．分散が大きいセンサほどユーザ要求を満たした復元が難しいため，これらセンサを割当の起点とすることで復元精度の全体的なバランスを取る狙いがある．次のステップでは，全センサの割当が決定するまで各スレッドへ 1 つずつ割り当てる（5–9 行目）．幅優先的に各スレッドへ順々に割り振ることでスレッド間のバランスを取っている．ただし，全ての割当パターンを検討するのは計算コストの観点から非現実的なため，各スレッドに対するセンサ割当は復元精度の評価関数に基づいて貪欲的に決定する（7 行目）．

行列計算が必要な 7 行目の処理が本アルゴリズムのボトルネックとなるため，ヒューリスティックスに基づいて更なる効率化を検討する．式 (11) の定義より式 (10) が小さくなるほど復元精度も向上する．ここで式 (10) の第 1 項は事前確率分布でありパーティショニングによる影響は無い．一方，式 (10) の第 2 項は他センサの処理結果を用いることによる分散の減少度合を表しており，復元対象のセンサとの相関が強いセンサを用いるほど減少度合も大きくなる．すなわち，相関の強いセンサを同じスレッドに多く割り当てるほど復元精度が向上する傾向にあるといえる．ただしエッジ障害を想定する場合，処理結果を損失したセンサと同一エッジへ割り当てられているセンサの処理結果も共に損失する点に注意が必要である．

以上より，同一エッジに割り当てられているセンサ同士の相関を 0 とした相関行列 R に対して，次の関数により復元精度を近似的に評価することでボトルネックの処理コストを軽減する．

$$f^*_{acc}\left(\boldsymbol{A}_i^0 \mid \mathcal{X}, \Sigma\right) = \sum_{a=0}^{n} \sum_{b=0}^{a} \mathbb{1}[X_a, X_b \in \boldsymbol{A}_i^0] R_{ab}^2 \tag{12}$$

ただし $\mathbb{1}[X_a, X_b \in \boldsymbol{A}_i^0]$ はセンサ X_a, X_b が共にエッジ i に割り当てられている場合 1 を，そうでない場合 0 を返す指示関数を表す．f^*_{acc} は単一エッジ障害時の復元に使用できるセンサ間の

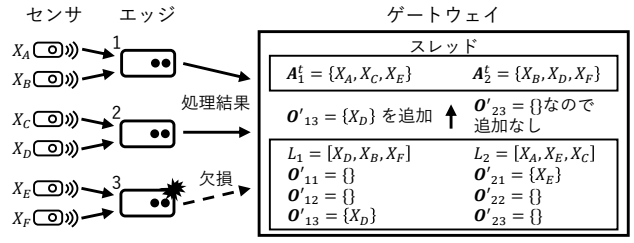

図 4　動的な割当制御の例

共分散の二乗和を表しており，別々のエッジに割り当てられている相関の強いセンサを同一スレッドに割り当てるほど評価値が大きくなる．そのためこの評価関数を用いることで，エッジ障害による影響の偏りを抑制しつつ，スレッド全体の推定精度を高い水準で保つようなパーティショニングが期待できる．

4.2　障害状況に応じた動的な割当制御

本節ではエッジの障害状況に応じた割当センサの動的な制御手法を提案する．2 節の議論より，近似的に復元した分散の値からユーザ要求を充足するかを判定できるため，基本的にはユーザ要求を充足するまで割当センサを追加すればよい．しかし，精度向上に効果のないセンサを 1 つずつ追加するのは非効率的なため，優先的に追加すべきセンサを事前に求めたい．

まず，スレッド i に対する追加候補のセンサ $\boldsymbol{X} \setminus \boldsymbol{A}_i^0$ を優先順に整列したリスト L_i を計算する．4.1 節の議論より相関が大きいセンサを割り当てるほど復元精度も向上するといえるため，評価関数 $f_{acc}^*(\boldsymbol{A}_i^0 \cup \{X\} \mid \mathcal{X}, \Sigma)$ の値が大きいセンサ X から順に L_i へ格納すればよい．例えば図 4 のスレッド 1 への追加候補のセンサは $\{X_B, X_D, X_F\}$ であり，センサ X_D の評価値が一番大きいとき L_1 の先頭に X_D を格納する．

次に，エッジ j に障害が発生したとき，スレッド i でユーザ要求を満たすため最初に確定で追加するセンサ集合 $\boldsymbol{O}'_{ij}$ を計算する．4.1 節の議論より，分散の目標値との差分 $\Delta\sigma_X$ が非負値となればユーザ要求を充足するといえる．よって，エッジ j の障害により処理結果を損失する全てのセンサ $\forall X \in \boldsymbol{A}_i^0 \cup \boldsymbol{X}_j$ について，$\boldsymbol{A}_i^0$ に加えたときの $\Delta\sigma_X$ が非負値となるまで L_i の先頭から順に $\boldsymbol{O}'_{ij}$ へ追加すればよい．例として図 4 のエッジ 3 に障害が発生したとき，スレッド 1 では $\{X_A, X_C\}$ の処理結果のみからユーザ要求を満たしてセンサ X_E の処理結果を復元できないとする．このとき，L_1 の先頭であるセンサ X_D を $\boldsymbol{O}'_{13}$ へ追加し，X_D を加えて復元する場合の $\Delta\sigma_{X_E}$ が非負値となるかを確かめる．$\Delta\sigma_{X_E}$ が非負値となるとき，それ以上のセンサは必要ないため $\boldsymbol{O}'_{13} = \{X_D\}$ を得る．

スレッド i は上述の事前計算した情報に基づき，エッジの障害状況に応じて割当センサを動的に制御する．エッジ j に障害が発生したとき，まずはじめに $\boldsymbol{O}'_{ij}$ を追加して損失した処理結果を復元する．ここで $\boldsymbol{O}'_{ij}$ は単一エッジ障害に対してユーザ要求を満たすために必要な最小限のセンサを表すため，エッジ j のみに障害が発生している場合はこれ以上のセンサの追加なしに復元結果を出力できる．しかし複数エッジに同時に障害が発生すると，対応する全ての $\boldsymbol{O}'_{ij}$ を追加するだけではユーザ要求を満たせない可能性がある．この場合，ユーザ要求を満たすまでリスト L_i の先頭から順にセンサを追加する．例えば図 4 においてエッジ 2,3 に障害が発生したとき，$\boldsymbol{O}'_{22}, \boldsymbol{O}'_{23}$ は共に空集合なためスレッド 2 ではセンサを追加せず X_B の結果から X_D, X_F の処理結果を復元する．この復元精度がユーザ要求を満たさないとき，L_2 の先頭であるセンサ X_A を追加して再度復元した結果を出力する．

障害状況やユーザ要求の厳しさ次第では，割当候補のセンサを全て追加してもユーザ要求を満たせない可能性がある点に注意する．このとき L_i の先頭から 1 つずつセンサを追加して復元結果がユーザ要求を満たすかを確かめるのは非効率的であり，各スレッドに対する割当センサ数が増えるほどこの計算コストも増加する．以上よりユーザ要求やシステムの構成機器の信頼度をふまえて，L_i から 1 度にまとめて追加するセンサ数および各スレッドへのセンサ割当数の上限を設定することで，復元精度と計算コストのトレードオフを制御できる．

5　評 価 実 験

本章では，処理遅延および復元精度に注目して本システムの性能を評価する．

5.1　実 験 設 定

a）　データセット

様々な規模での提案システムの性能を評価するため，多次元ガウス分布に従う人工データ 2880 タプルを作成した．全 n 個のセンサを各エッジに対して 20 個ずつ割り当て，分散は全て 1 に設定し，期待値は同一エッジで共通の値とした上で時間経過と共に変動させた．また，同一エッジに割り当てたセンサ同士には強い相関を，別エッジに割り当てたセンサ同士には原則として弱い相関をそれぞれ設定した．ただし，パーティショニングアルゴリズムの効果を見るため，別エッジに割り当てた一部のセンサ同士にも強い相関を設定した．センサ数の規定値は $n = 500$ とし，データ生成時に使用したモデルをそのまま実験に使用する．

b）　実験シナリオ

本実験ではセンサ，エッジ，およびゲートウェイから構成されるシステムを想定する．各センサは一定間隔で計測値を割当エッジに対して送信し，エッジでは窓幅 $w = 10$，滑り幅 $l = 2$ の設定のもと各センサの平均値を計算する．そして，ゲートウェイでは障害による欠損値を復元し，エッジの処理結果をまとめて出力する．ただし各エッジに対して 20 個のセンサを，ゲートウェイの各スレッドに対して 50 個のセンサをそれぞれ割り当てるものとする．

この構成において，エッジ故障を模倣した障害シナリオに基づき各機器の処理状況をシミュレートする．各エッジは運転状態から故障状態へ確率 r_f で遷移すると，これまで受け取った入力および内部状態を削除し，故障状態にある間は一切の動作を停止する．そして故障状態に遷移してから t_f 単位時刻後に運転状態へと復帰すると，データ処理や出力などの処理を再開

表 1　実験用サーバの構成

機器名	Dell PowerEdge R640
OS	Ubuntu 20.04.5 LTS
CPU	Intel(R) Xeon(R) Gold 6262V CPU @ 1.90GHz
メモリ	256GB
ストレージ	480GB

する．なお，規定値はそれぞれ $r_f = 0.01$, $t_f = 10$ とする．

また，データソースには実際のセンサではなくシミュレータを使用する．実際の計測間隔では実験に時間がかかりすぎるため，使用データセットそれぞれに対して設定した単位時刻 $T = 0.1[s]$ 毎にシミュレータから対応するエッジへと計測値を送信する．エッジやゲートウェイの処理間隔もシミュレータからのデータ送信間隔に応じてそれぞれ設定する．

c）実　　装

本システムの性能評価実験のためプロトタイプを構築した．エッジでは欠損値の復元とデータ集約を，ゲートウェイでは欠損値の復元をそれぞれマイクロバッチ形式で実行する．エッジやゲートウェイの入力キューとして Kafka [10] などのメッセージキューが一般的であるが，本実験では特定時刻のデータ削除などの障害シナリオに従った細かな制御が容易な PostgreSQL を採用した．環境センサとの通信制御などが必要なエッジ環境への適応性を鑑みて Python 3.9.0 を用いて実装しており，行列演算は全て NumPy を用いて計算する．

本実験ではデータソースのシミュレータも含め全てのプログラムを 1 台のサーバ上で実行した．実験に使用したマシンの構成を表 1 に示す．エッジデバイスの実機を用いた性能評価は今後の課題とする．

d）比 較 対 象

以上の設定のもと，以下の 5 手法の実験結果を比較することで提案システムの各機能が及ぼす効果を評価する．

- **single**：1 スレッドのみで処理する場合の実験結果．
- **random**：ランダムに初期割当を決定した場合の実験結果．
- **vanilla**：図 3 のアルゴリズムに基づいて初期割当を決定した場合の実験結果．
- **proposed**：図 3 のアルゴリズムにおいて復元精度を式 (12) に基づいて評価して初期割当を決定した場合の実験結果．
- **proposed+**：proposed の初期割当から障害状況に応じて動的に割当を制御する場合の実験結果．ただし誤差要求の規定値を $\delta = 0.9$，$\epsilon = 0.25$ とし，各スレッドの割当数の上限を 150 とした．

5.2　処理精度の評価

理論的に求められる誤差上限およびユーザ要求の充足率の 2 つの指標により本システムの復元精度を評価する．各センサ $X \in \boldsymbol{X}$ の全ての復元結果に対して誤差上限 e_X を計算し，その平均二乗和を理論的な誤差量 $rmse_X$ として算出する．また充足率は各復元結果の誤差上限 e_X がユーザ指定の許容誤差 ϵ 以下に収まる割合を指すものとする．ここで誤差上限 e_X はユーザ指定の要求信頼度 δ に対する誤差幅であり，充足率が 1 となったとき全てのデータをユーザ要求を満たして復元できたといえる．ただし，障害による欠損が発生しなかったセンサの処理結果は誤差ゼロのため評価対象から除く．

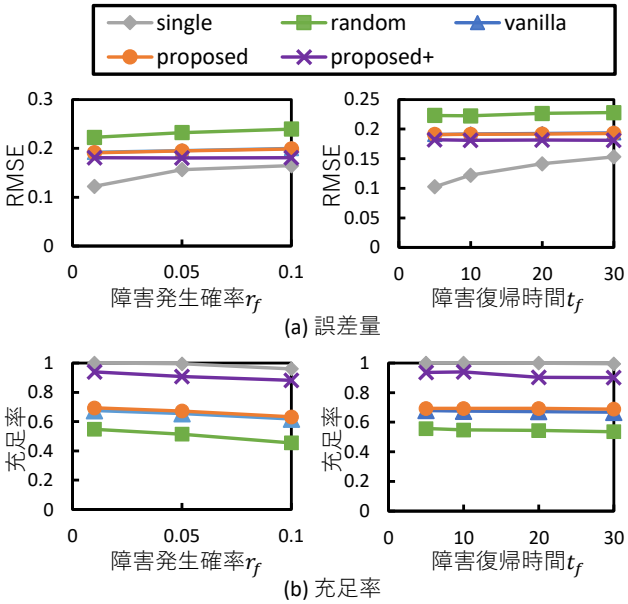

図 5　欠損シナリオに対する RMSE の変化

欠損シナリオのパラメータである障害発生確率 r_f および障害復帰時間 t_f を変化させたときの実験結果を図 5 に示す．実験結果より，proposed は random に比べ誤差量と充足率をともに改善できていることから，初期割当を工夫することで復元精度を改善できるとわかる．また proposed は vanilla と同等の復元精度を達成できており，ヒューリスティックスに基づいて設計した近似的な評価関数の妥当性が見て取れる．一方，proposed+ の誤差量は proposed のものと大きく変わらないが，充足率を大幅に改善できている．このことから，提案手法の動的制御では復元結果がユーザ要求を充足できないときにのみ，必要最小限のセンサを追加で割り当てられているとわかる．

5.3　処理遅延の評価

応答時間に基づいて本システムの処理遅延を評価する．各時間窓内に割り当てられた計測値の内最小の送信時刻を入力時刻 t_i，その時間窓の処理結果がゲートウェイからシンクへ出力された時刻を出力時刻 t_o とそれぞれ定義し，応答時間はこれら時刻の差 $t_o - t_i$ として求める．

欠損シナリオのパラメータである障害発生確率 r_f および障害復帰時間 t_f を変化させたときの実験結果を図 6 に示す．なお応答時間はデータの送信間隔や窓幅に大きく依存することから，一切の欠損がないときの理想的な応答時間を参考値として破線で示している．

実験結果より，single ではデータの欠損が増えるほど応答時間も増加するのに対して，proposed の応答時間はパラメータの変化によらずほとんど理想的な値となっている．また，proposed の応答時間は vanilla や random と変わらないことから，初期割当の違いによる応答時間への影響は無いとわかる．以上より，シングルスレッドでは計算コストの重さから処理遅延が増大す

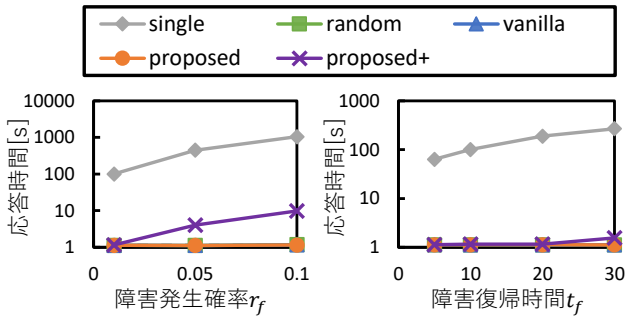

図 6　欠損シナリオに対する応答時間の変化

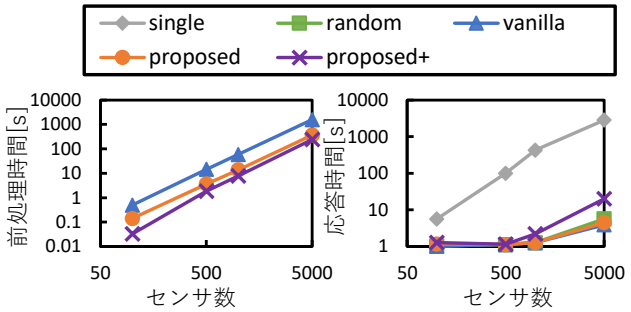

図 7　システム規模に対する前処理時間および応答時間の変化

るような環境下でも，並列処理により処理遅延を大幅に削減できるといえる．一方，proposed+は欠損が増えるほど応答時間が増加しているが，その増加幅は小さく抑えられている．特に障害発生確率 $r_f = 0.1$ のとき全体の約半数のデータが欠損することになるが，proposed+は single に比べて処理遅延を 100 倍以上削減しながら，90% 以上の充足率を達成できている．以上より，提案手法では障害状況に応じて必要最低限のセンサを追加することで，ユーザの誤差要求を満たせる範囲で処理遅延を大きく削減できているとわかる．

proposed+の結果について詳しく見ると，障害復帰時間 t_f よりも障害発生確率 r_f の増加に対して応答時間の増加が大きくなる傾向にある．これは，障害発生確率 r_f の増加により同時刻に多くのエッジに障害が発生しやすくなり，追加候補のセンサ集合 $\boldsymbol{O}'_{ij}$ だけの追加ではユーザ要求を満たせないケースが増えてしまったためであると考えられる．複数エッジの障害に対してもより高速にユーザ要求を満たす割当の決定が今後の課題である．

次に、システム規模に対する性能変化をみるため，センサ数 n を変化させたときの応答時間および前処理の所要時間を図 7 に示す．ただし，前処理の所要時間として proposed および vanilla は初期割当決定に要した時間を，proposed+は割当優先順のリスト L_i および追加候補のセンサ集合 $\boldsymbol{O}'_{ij}$ の計算時間をそれぞれ示す．

実験結果より，センサ数の増加に対して前処理の所要時間も増加しているものの，vanilla に比べて proposed は 5 倍ほど計算時間を短縮できている．また proposed+の前処理時間も初期割当決定に要する時間以内で収まっており，近似的な評価関数により前処理での計算コストも削減できているといえる．

次に応答時間について見ると，single はセンサ数の増加に対して応答時間も増加傾向にあるが，proposed の応答時間はほとんど理想的な値になっている．特にセンサ数 $n = 5000$ のとき，porposed は single に比べて処理遅延を 1000 倍以上短縮できている．また，proposed+の応答時間は増加傾向にあるものの増加幅は小さく抑えられており，single に比べて処理遅延を最大で 100 倍以上短縮できている．以上より提案手法ではシステム規模が増加したとしても，ユーザの誤差要求を満たせる範囲で処理遅延を大きく削減できるとわかる．

6　関 連 研 究

本章ではデータストリーム処理システムにおける耐障害性保証およびパーティショニング制御の関連研究について概説する．

6.1　耐障害性保証

既存システムの多くは耐障害性保証として at-most-once セマンティクスや at-least-once セマンティクスを提供する [6, 7]. at-most-once セマンティクスを保証するシステムでは内部状態などのバックアップを取らず待機ノードへの処理移譲のみを行うため，データ損失の恐れがある．一方，at-least-once セマンティクスを保証するシステムでは障害復帰処理として入力データを再送するが，同一データを複数回処理する可能性がある．つまり，これらのセマンティクスに基づくシステムでは障害により生じる誤差量を保証できない．

Flink [4] や Spark Streaming [5] などのシステムでは，誤差のない頑健な耐障害性保証として exactly-once セマンティクスを提供する．これらのシステムでは内部状態の定期的なバックアップおよび復旧時の入力データの適切な再処理により，障害発生時も入力されたデータを過不足なく 1 回処理した結果が得られる．ただし，他のセマンティクスに比べバックアップの取得手順は複雑であり，処理のスループットも低下する．

また Huang らは耐障害性保証におけるコスト削減のため，結果の信頼性を保証した近似的な耐障害性保証を提案した [11]. 未処理データ数と内部状態の変量に対するしきい値を超した場合にのみバックアップを取ることで，バックアップデータ量およびその頻度を削減できスループットが向上する．

これら従来型の耐障害性保証は主にクラウド環境を想定して設計されており，エッジ環境への適用には問題がある．エッジ環境ではデバイス故障に対して人手で修理ないし交換する必要があるため，クラウド環境のような仮想マシンの立ち上げによる早急な障害復帰は望めない．またエッジ環境特有の物理的制約の強さから，システム規模が大きくなるほどデバイスの多重化コストは増加し，かつ代替マシン活用の柔軟性もクラウド環境に劣る．すなわち，上述の待機系への処理移譲を基本とする従来アプローチでは障害発生時の処理継続が困難であるといえる．

この課題に対して，我々のグループでは近似的な耐障害性保証に基づくエッジストリーム処理システムを提案した [2,3]. 前段機器の障害により損失したデータを後段の機器で近似的に復元することで，エッジ側での物理的な多重化や処理の途中結果

の定期的なチェックポインティングなしに，故障機器の復旧を待たずに処理を継続できる．そのため障害によるダウンタイムが長期化しがちなエッジ環境においても，処理遅延と可用性を高水準に維持できる．

6.2 パーティショニング制御

既存システムの多くは大規模ストリームを高速に処理するため，キーパーティショニングに基づく並列処理をサポートしている [4–6,12,13]．データストリーム処理システムでは一般的に，各スレッドに対して入力データをより均等に割り振るほど処理性能も向上することから，パーティショニングアルゴリズムはシステムの処理性能を決定する重要な要素の 1 つである [14]．

特に集約処理などの内部状態を伴うステートフル処理では，キーの出現頻度の偏りやワークロードの変動の影響を大きく受けることから，効率的な処理を実現するための様々な手法が提案されている [14–17]．例えば Katsipoulakis らの手法 [14] は複数スレッドで並列に集約した結果を後で 1 つにまとめる二段階の集約処理を想定し，全体的な計算コストを考慮してキー分割することで，単純なシャッフリングやハッシュに基づく手法に比べて 10 倍以上高速に処理できる．また Zapridou らの手法 [17] は，強化学習を活用しワークロードの変化に追従してパーティショニングポリシーを動的に制御することで，他の手法に比べて 6.7 倍高速に処理できる．しかし近似的な耐障害性保証のアプローチでは，キーの偏りやワークロード変化ではなくパーティショニングによる処理精度低下が課題であるため，これら既存手法は復元精度と処理性能のトレードオフの問題解決には適さない．

7 まとめ

本稿では近似的な耐障害性保証に基づくデータストリーム処理を想定し，復元精度と処理性能のトレードオフを考慮したパーティショニング手法を提案した．本手法はヒューリスティックスを活用した初期割当アルゴリズムおよび，障害発生状況に応じた動的な割当制御により，全てのセンサの処理結果をユーザ要求を満たして復元できる範囲で計算コストを最小化する．実験により，提案手法は障害状況に対して必要最小限のセンサを追加することで，ユーザの誤差要求を高い確率で満たしながら処理遅延をシングルスレッドに比べて 100 倍以上削減できることを確認した．

今後の課題として，誤差要求と計算コストのトレードオフをユーザが対話的に調整できるなど，運用面まで考慮したアルゴリズムの改善が挙げられる．また，提案手法の各工夫点の効果をより精密に評価するとともに，実データやセンサ故障の考慮などより現実的な設定での評価実験についても取り組む予定である．

謝辞

本研究は JSPS 科研費 JP20K19804，JP21H03555，JP22H03594 の助成，および国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務（JPNP16007）の結果得られたものである．

文献

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[2] 高尾大樹, 杉浦健人, 石川佳治, “エッジコンピューティングにおける低遅延かつ高信頼度なデータストリームの近似的集約処理,” **電子情報通信学会論文誌** *D*, vol. J104-D, no. 5, pp. 463–475, 2021.

[3] D. Takao, K. Sugiura, and Y. Ishikawa, “Approximate fault tolerance for edge stream processing,” in *Database and Expert Systems Applications - DEXA 2021 Workshops*, pp. 173—183, 2021.

[4] “Apache Flink: Stateful Computations over Data Streams.” `https://flink.apache.org/` (Accessed: Jan. 8, 2022).

[5] “Spark Streaming | Apache Spark.” `https://spark.apache.org/streaming/` (Accessed: Jan. 8, 2022).

[6] “Apache Storm.” `https://storm.apache.org/` (Accessed: Jan. 8, 2022).

[7] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, “S4: Distributed stream computing platform,” in *2010 IEEE Int. Conf. on Data Mining Workshops*, pp. 170–177, 2010.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] D. C. Lay, S. R. Lay, and J. J. McDonald, *Linear Algebra and Its Applications*. Pearson, 5th ed., 2014.

[10] “Apache Kafka.” `https://kafka.apache.org/` (Accessed: Jan. 8, 2022).

[11] Q. Huang and P. P. C. Lee, “Toward high-performance distributed stream processing via approximate fault tolerance,” *Proc. VLDB*, vol. 10, no. 3, pp. 73–84, 2016.

[12] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. Fernández-Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt, *et al.*, “The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing,” *Proc. VLDB*, vol. 8, pp. 1792–1803, 2015.

[13] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica, “Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters,” in *Proc. the 4th USENIX Conf. on Hot topics in Cloud Computing*, p. 10, 2012.

[14] N. R. Katsipoulakis, A. Labrinidis, and P. K. Chrysanthis, “A holistic view of stream partitioning costs,” *Proc. VLDB*, vol. 10, no. 11, pp. 1286–1297, 2017.

[15] M. Anis Uddin Nasir, G. De Francisci Morales, D. Garcia-Soriano, N. Kourtellis, and M. Serafini, “The power of both choices: Practical load balancing for distributed stream processing engines,” in *Proc. ICDE*, pp. 137–148, 2015.

[16] M. Anis Uddin Nasir, G. De Francisci Morales, N. Kourtellis, and M. Serafini, “When two choices are not enough: Balancing at scale in distributed stream processing,” in *Proc. ICDE*, pp. 589–600, 2016.

[17] E. Zapridou, I. Mytilinis, and A. Ailamaki, “Dalton: Learned partitioning for distributed data streams,” *Proc. VLDB*, vol. 16, no. 3, pp. 491–504, 2023.