DEIM2022 D23-4

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の開発

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において耐障害性保証は重要な課題の 1 つである．従来手法では，待機系への処理移譲および損失データの再処理のために多数の機器との協調が必要となり，処理遅延や可用性を損なう恐れがある．この問題に対して著者らは，近似的な耐障害性保証のアプローチを提案した．各障害によるデータ損失を後段の機器で近似的に復元することで，システム全体として障害によらず処理を継続でき，処理遅延や可用性の改善が期待できる．しかし，これまで理論的な妥当性および復元精度しか評価されておらず，システムとしての性能は未だ未知数である．そこで本稿では，耐障害性を近似的に保証するデータストリーム処理システムを構築し，その処理遅延および可用性を実験により評価する．また，近似的な耐障害性保証において効果的なウォーターマークの制御方法についても提案する．

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，高可用性．

1 はじめに

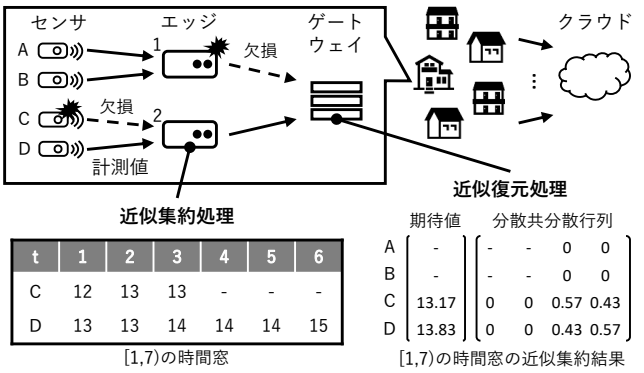
分散データストリーム処理システムにおいて耐障害性保証は重要な要件であり，これまで様々なアプローチが提案されている [1–4]．Flink [1] など多くのシステムでは複数タスクを介したパイプライン処理を採用しているが，このような処理形態ではあるタスクからの出力が途絶えると以降全てのタスクでの処理も停滞してしまう．システムの処理遅延や可用性を維持するためには，障害が発生しても処理を継続する仕組みが必要である．既存システムでは一般的に待機系への処理移譲，および損失データの再処理によって頑健な耐障害性を保証する．すなわちあるノードの障害を検知した際には待機ノードがその処理内容を引き継ぎ，また障害によって損失したデータを再処理することで，誤差なく処理を継続する．

しかし，従来の耐障害性保証アプローチは主にクラウド環境を想定したものであり，エッジコンピューティング環境ではシステム性能を損なう恐れがある．エッジコンピューティングとは，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施すことで，データ通信量の削減や処理の負荷分散を可能とする処理形態である [5]．例としてスマートシティでの環境センシングを考える．各区画や建物内に広く設置された環境センサデバイスは計測データを無線通信によってそれぞれ近くのエッジデバイスへと送信しする．そして各エッジは一定時間ごとに計測データを集約し，ゲートウェイマシンへ転送する．このようなシナリオにおいて，従来型の耐障害性保証には以下の問題点が挙げられる．

- **可用性の低下**：多数の安価なセンサやエッジ（以降本稿ではデバイスと総称する）を用いるエッジコンピューティング環境では，クラウド環境に比べ障害の発生確率が高くなると考えられる．そのため，障害復帰処理に必要なシステム全体の処理中断の影響も大きくなり，可用性が低下してしまうと考えられる．
- **処理遅延の増大**：障害検出や再処理のためにはゲートウェイおよび多数のエッジ間で協調した処理が必要であるため，クラウド環境とは異なりリソースの集中管理が困難なエッジ環境ではオーバヘッドが増大すると考えられる．
- **コストパフォーマンスの低下**：頑健な耐障害性保証のためにはエッジ単位での多重化が必要であるが，仮想マシンなどで代替ノードをすぐに立ち上げられるクラウド環境と異なり，そのコストは無視できない．また，多重化したとしてもセンサ故障や通信障害によるデータ欠損は避けられず，頑健な耐障害性保証のメリットは小さい．

この問題に対して著者らは，近似的な耐障害性保証のアプローチを提案した [6,7]．本手法は図 1 に示すとおり，エッジでの近似集約処理および，ゲートウェイでの近似復元処理の大きく 2 つの処理から構成される．図 1 は，各エッジは窓幅 $w = 6$ での各センサの平均値を計算し，処理結果をゲートウェイへと送信する状況を表している．ただし，各デバイスの障害により “-” で示した一部の計測値および処理結果が損失したとする．このとき，エッジ 2 では得られた計測値の情報を活用することで，故障したセンサ C の集約結果を近似的に計算する．また，ゲートウェイではエッジ 2 から得られるセンサ C・D の集約結果を用いて，エッジ 1 の故障により損失したセンサ A・B の集約結果を近似的に復元する．本手法では各エッジでの近似処理による不確実性の情報を確率分布として表現し，この不確実性の情報を考慮して障害エッジ出力を復元することで誤差上限を正しく推定できる．上述のとおりデバイス単位で多重化をせずとも障害の発生によらず処理を継続できるため，本手法により処理遅延や可用性，コストパフォーマンスの改善が期待できる．しかし，これまで本手法の理論的な妥当性および近似処理精度しか評価されておらず，処理遅延や可用性の評価は十分でない．

そこで本稿では，耐障害性を近似的に保証するストリーム処理システムを構築し，その処理遅延および可用性を実験により



t	1	2	3	4	5	6
C	12	13	13	-	-	-
D	13	13	14	14	14	15

	期待値	分散共分散行列			
A	-	-	-	0	0
B	-	-	-	0	0
C	13.17	0	0	0.57	0.43
D	13.83	0	0	0.43	0.57

図 1　近似的耐障害性保証アプローチの概要図

評価する．エッジおよびゲートウェイでの具体的な処理手順や永続化が必要なデータの種類などシステムの詳細を設計する．また，提案アプローチにおける誤差保証の仕組みを最大限活かしたウォータマークの制御方法の改善についても提案する．そして，プロトタイプシステムを構築し，提案システムの処理遅延および可用性を実験により評価する．

本稿の構成は以下のとおりである．2 章では，ストリーム処理システムにおける耐障害性保証に関する先行研究について概説する．また，3 章では近似的な耐障害性保証について，4 章では耐障害性を近似的に保証するストリーム処理システムの設計についてそれぞれ説明する．そして，5 章では提案システムの処理遅延および可用性に関する評価実験について述べる．最後に，本研究のまとめと今後の課題を 6 章で述べる．

2　関 連 研 究

従来のストリーム処理システムの多くは耐障害性保証セマンティクスとして，at-most-once セマンティクスや at-least-once セマンティクスを保証する [3,8]．at-most-once セマンティクスを保証するシステムでは，内部状態やデータ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，データの損失の恐れがある．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．これらセマンティクスを保証するシステムでは，障害により生じる誤差量を保証できないため信頼性に問題がある．

また，Flink [1] や Spark Streaming [2] などのストリーム処理システムでは，誤差のない頑健な耐障害性保証として exactly-once セマンティクスを保証できる．これらシステムでは，内部状態の定期的なバックアップおよび入力データの再処理に基づく復旧により，障害の発生によらず入力されたデータを過不足なく 1 回処理した結果の出力を保証する．クラウド環境に注目した場合，豊富なリソースが物理的に集中しておりまたシステム前段のメッセージングキューに入力データの管理を一任できるため，この耐障害性保証のアプローチはコストと信頼性の両面において優れている．

しかし，これら従来手法は主にクラウド環境を想定して設計されており，エッジコンピューティング環境へ適用する場合様々な問題がある．具体的な障害復帰方法に違いはあるものの，従来の耐障害性保証アプローチは待機系への処理移譲を基本としている．エッジコンピューティング環境ではその物理的な制約からエッジ単位での物理的な多重化が必要となり，システム規模が大きくなるほど多重化コストも増加する．また障害復帰に入力データの再処理が必要な場合，再処理中にも次々と新しい入力が生成されるため，データ生成時刻から実際の処理完了時刻までの遅延が増大する問題もある．クラスタ環境においてはこの遅延を抑制するために障害復帰処理を並列化するアプローチ [9] などが提案されているが，物理的多重化コストが無視できないエッジ環境においては異なるアプローチが求められる．

耐障害性保証におけるコストの問題に対して，結果の信頼性を保証した近似的な耐障害性保証のアプローチ [10] がある．この手法は未処理データ数と内部状態の変量に対する閾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しスループットを向上する．しかし，この手法は計算コストの問題にのみ着目しており待機系への処理移譲が必須なため，物理的多重化コストの問題は解決できない．

3　近似的な耐障害性保証

本章では，エッジコンピューティング環境を想定した近似的な耐障害性保証アプローチについて説明する．図 1 に示すとおり，本手法はエッジでの近似集約処理，およびゲートウェイでの近似復元処理の大きく 2 つの処理から構成される．本手法では，前段機器の障害により損失したデータを後段機器で近似的に復元することで，障害デバイスの復旧を待たずシステム全体として処理を継続する．また，本手法では近似集約結果を全て確率分布として計算することで，各エッジでの近似処理による不確実性を考慮して処理結果の誤差上限を理論的に保証する．以降では，エッジでの近似集約処理，およびゲートウェイでの近似復元処理についてそれぞれ述べる．

3.1　エッジでの近似集約処理

エッジでは，センサから受け取った計測値系列を時間窓により分割し，時間窓ごとに各センサの集約値を計算する．時刻 $t \in \mathbb{N}^+$ における n 台のセンサ $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$ の計測値の真値を $\boldsymbol{x}^t = \langle x_1^t, x_2^t, \ldots, x_n^t \rangle$ と表す．エッジ i では，時間窓の始点 t'，および窓幅 w に対して，センサ $\boldsymbol{X}_i \subseteq \boldsymbol{X}$ の計測値系列 $\boldsymbol{x}_i^{[t',t'+w)} = \langle \boldsymbol{x}_i^{t'}, \boldsymbol{x}_i^{t'+1}, \ldots, \boldsymbol{x}_i^{t'+w-1} \rangle$ から各センサ $X \in \boldsymbol{X}_i$ の集約値 $\boldsymbol{Y}_X$ を計算する．ただし，本稿では集約問合せとして平均値および合計値問合せを想定する．

センサ故障や通信障害などの問題に対して，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，得られた一部の計測値から欠損値を確率分布の形で推定する．本手法では，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 $\boldsymbol{X}$ 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

センサの計測値 $\boldsymbol{X}^{t+1}$ は時刻 t の計測値 $\boldsymbol{X}^t$ 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，本手法は以下の 3 つのステップで各センサの集約結果を近似的に計算する．

（1） 各エッジは時刻 t で受け取った一部のセンサの計測値および事前確率分布を用いて，エッジに割り当てられている全センサの事後確率分布を計算する．

（2） 遷移モデルおよびステップ 1 で求めた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．

（3） 上述の 2 ステップを各時刻で反復的に繰り返し得られた事後確率分布の系列を時間窓で集約する．

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 での事前確率分布が正規分布 $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ として与えられたとき，時刻 t でエッジ i が受け取ったセンサ $\boldsymbol{O}_i^t \subseteq \boldsymbol{X}_i$ の計測値 $\boldsymbol{o}_i^t$ からセンサ $\boldsymbol{Z}_i^t = \boldsymbol{X}_i \setminus \boldsymbol{O}_i^t$ の事後確率を表す正規分布 $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Z}_i^t|\boldsymbol{o}_i^t}, \Sigma_{\boldsymbol{Z}_i^t|\boldsymbol{o}_i^t})$ を次のとおり求める [11].

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

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

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（つまり $\boldsymbol{Z}_i^t$ 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（つまり $\boldsymbol{Z}_i^t$ と $\boldsymbol{O}_i^t$ の共分散を表すブロック行列）である．またこのとき，$\boldsymbol{X}_i^t$ の事後確率分布を $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t}, \Sigma_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t})$ として表す．ただし，センサ $\boldsymbol{O}_i^t$ に対応する期待値は $\boldsymbol{o}_i^t$ とし，分散および他のセンサとの共分散は全て 0 とする．

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．

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

特に時刻 $t+1$ および t の計測値の同時確率分布 $P(\boldsymbol{X}_i^{t+1}, \boldsymbol{X}_i^t)$ 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 $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{X}_i^{t+1}|\boldsymbol{o}_i^t}, \Sigma_{\boldsymbol{X}_i^{t+1}|\boldsymbol{o}_i^t})$ が得られる．

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

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

ただし，$\boldsymbol{\mu}''$，および Σ'' の添字について A は先の時刻の計測値に対応する次元を，B は前の時刻の計測値に対応する次元をそれぞれ抽出することを示す．

例として図 1 を考えると，時刻 3 においてエッジ 2 はセンサ C，D の計測値を得ているため，$\boldsymbol{\mu}_{\boldsymbol{X}_2^3|\boldsymbol{o}_2^3} = \boldsymbol{o}_2^3$ であり，$\Sigma_{\boldsymbol{X}_2^3|\boldsymbol{o}_2^3}$ の全要素は 0 となる．この事後確率分布を式 (4)，(5) に代入することで，時刻 4 における事前確率分布 $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{X}_2^4|\boldsymbol{o}_2^3}, \Sigma_{\boldsymbol{X}_2^4|\boldsymbol{o}_2^3})$ が得られる．こうして得られた事前確率分布を用いることで，時刻 4 のセンサ C の欠損値を精度よく推定できる．

以上の処理を反復的に繰り返すことで計測値間の相関関係を考慮しながら欠損値を近似的に復元できるため，あとはこれを時間窓で集約すればよい．例えば，センサ $X \in \boldsymbol{X}_i$ の平均値 $Y_X = (\sum_{t \in [t', t'+w)} X^t)/w$ に対して，各時刻の期待値ベクトル $\boldsymbol{\mu}_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t}$ および分散共分散行列 $\Sigma_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t}$ から次の正規分布 $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}_{\boldsymbol{X}_i}}, \Sigma_{\boldsymbol{Y}_{\boldsymbol{X}_i}})$ が得られる [6].

$$\boldsymbol{\mu}_{\boldsymbol{Y}_{\boldsymbol{X}_i}} = \frac{1}{w}\left(\sum_{t \in [t', t'+w)} \boldsymbol{\mu}_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t}\right) \tag{6}$$

$$\Sigma_{\boldsymbol{Y}_{\boldsymbol{X}_i}} = \frac{1}{w^2}\left(\sum_{t \in [t', t'+w)} \Sigma_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t}\right) \tag{7}$$

集約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 $X \in \boldsymbol{X}_i$ の平均値 Y_X が μ_{Y_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．

$$\begin{aligned} &P\left(Y_X \in [\mu_{Y_X} - e, \mu_{Y_X} + e]\right) \\ &= \int_{\mu_{Y_X}-e}^{\mu_{Y_X}+e} f\left(y \mid \mu_{Y_X}, \Sigma_{Y_X}\right) dy \end{aligned} \tag{8}$$

ここで，$f(y \mid \mu_{Y_X}, \Sigma_{Y_X})$ は正規分布 $\mathcal{N}(\mu_{Y_X}, \Sigma_{Y_X})$ に対する確率密度関数を表す．式 (8) より，ユーザ要求として要求信頼度 δ が与えられたとき，$P(Y_X \in [\mu_{Y_X} - e', \mu_{Y_X} + e']) = \delta$ を解くことで，要求信頼度 δ を満たす最小の推定誤差 e' を計算できる．このとき，δ の確率で Y_X の真値が $\mu_{Y_X} \pm e'$ 以内に存在することを保証する．ただし，$\Sigma_{Y_X} = 0$ のときは時間窓中の全時刻において計測値が得られているため，常に $e' = 0$ であり集約結果に誤差は含まれない．

3.2 ゲートウェイでの近似復元処理

ゲートウェイでも時間的相関およびセンサ間の相関を考慮し，エッジ障害により損失した処理結果を近似的に復元する．障害エッジに割り当てられたセンサ $\boldsymbol{Z} \subseteq \boldsymbol{X}$ の集約結果は全て損失し，センサ $\boldsymbol{O} = \boldsymbol{X} \setminus \boldsymbol{Z}$ から集約結果として $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}_O}, \Sigma_{\boldsymbol{Y}_O})$ が得られるとする．ただし，$\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}_O}, \Sigma_{\boldsymbol{Y}_O})$ は複数のエッジからの出力を合わせたものである点に注意する．エッジ $i, j (i \neq j)$ はそれぞれが独立して近似集約処理を行っているため，本稿では $\boldsymbol{Y}_{\boldsymbol{X}_i}$，$\boldsymbol{Y}_{\boldsymbol{X}_j}$ 間の共分散を表す要素は全て 0 として扱う．

例として図 1 を考えると，ゲートウェイは $[1, 7)$ の時間窓に対してエッジ 1 の故障によりセンサ $\boldsymbol{Z} = \{A, B\}$ の集約結果を損失しており，センサ $\boldsymbol{O} = \{C, D\}$ の集約結果として正規分布 $\mathcal{N}(\mu_{\boldsymbol{Y}_O}, \Sigma_{\boldsymbol{Y}_O})$ のみを得ている．なお，エッジ 1 と 2 はそれぞれ独立して処理しているため，センサ $\boldsymbol{Z}$ および $\boldsymbol{O}$ 間の共分散は全て 0 としている．

センサ $\boldsymbol{Z}$ の集約結果 $\boldsymbol{Y}_{\boldsymbol{Z}}$ に対する事後確率分布は次のとおり計算できる．

$$P\left(\boldsymbol{Y}_{\boldsymbol{Z}}\right) = \int P\left(\boldsymbol{Y}_{\boldsymbol{Z}} \mid \boldsymbol{Y}_{\boldsymbol{O}} = \boldsymbol{y}\right) P\left(\boldsymbol{Y}_{\boldsymbol{O}} = \boldsymbol{y}\right) d\boldsymbol{y} \tag{9}$$

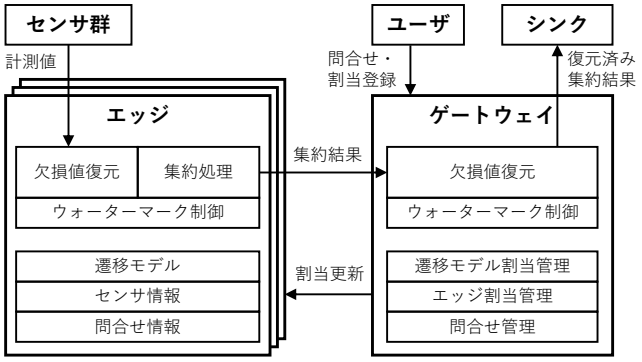


図 2　提案システムにおける処理の流れ

ここで，集約結果 $\boldsymbol{Y_X}$ 間の相関関係を正規分布 $\mathcal{N}(\boldsymbol{\mu'}, \Sigma')$ により表現できるとき，式 (9) を計算することで $\boldsymbol{Y_Z}$ の事後確率として次の正規分布 $\mathcal{N}\left(\boldsymbol{\mu_{Y_Z|Y_O}}, \Sigma_{\boldsymbol{Y_Z|Y_O}}\right)$ が得られる [7].

$$\boldsymbol{\mu_{Y_Z|Y_O}} = \boldsymbol{\mu'_Z} + \Sigma'_{\boldsymbol{ZO}} \left(\Sigma'_{\boldsymbol{OO}}\right)^{-1} \left(\boldsymbol{\mu_{Y_O}} - \boldsymbol{\mu'_O}\right) \tag{10}$$

$$\begin{aligned}\Sigma_{\boldsymbol{Y_Z|Y_O}} =& \Sigma'_{\boldsymbol{ZZ}} - \Sigma'_{\boldsymbol{ZO}} \left(\Sigma'_{\boldsymbol{OO}}\right)^{-1} \Sigma'_{\boldsymbol{OZ}} \\ &+ \Sigma'_{\boldsymbol{ZO}} \left(\Sigma'_{\boldsymbol{OO}}\right)^{-1} \Sigma_{\boldsymbol{Y_O}} \left(\Sigma'_{\boldsymbol{OO}}\right)^{-1} \Sigma'_{\boldsymbol{OZ}}\end{aligned} \tag{11}$$

ここで，各エッジでの近似処理による不確実性は $\Sigma_{\boldsymbol{Y_O}}$ によって表されており，式 (11) の第 3 項によってこれを考慮して分散の値を補正することで誤差上限を正しく保証する．

事前確率分布および $\mathcal{N}(\mu_{\boldsymbol{Y_O}}, \Sigma_{\boldsymbol{Y_O}})$ から $\mathcal{N}(\mu_{\boldsymbol{Y_Z}}, \Sigma_{\boldsymbol{Y_Z}})$ を近似的に復元できれば，次の時間窓に対する事前確率分布の計算は 3.1 節と同様の議論ができるため，エッジ 1 の復旧を待たずゲートウェイの処理を継続できる．

4　システム設計

本章では，耐障害性を近似的に保証するストリーム処理システムの設計について説明する．4.1 節では提案システムの構成および具体的な処理手順について述べる．また，提案アプローチにおける誤差保証の仕組みを活かした効率的なウォーターマーク制御手法を 4.2 節で述べる．

4.1　提案システムの概要

提案システムの構成および処理手順を図 2 に示す．本システムでは集約問合せを想定しており，各エッジでの近似集約結果をゲートウェイで取り纏め，クラウドサーバなど後段のシンクへと送信する．デバイスの故障や通信障害をはじめとする各障害によるデータ損失の問題に対して，本システムではその後段の機器で近似的に復元することで，欠損のない近似的な集約結果を誤差保証と共に絶えずリアルタイムに提供する．以降では，本システムにおける処理の流れの詳細を説明する．ただし，ウォーターマークの制御方法については 4.2 節で述べる．

まずはじめに，ユーザはゲートウェイに対して問合せ情報および処理に必要な各種割当情報を登録・更新する．ゲートウェイではこれら登録された情報に基づき，各エッジに対して遷移モデルやセンサ情報，問合せ情報を送信する．このときエッジでの計算量およびデータ量削減のため，ゲートウェイで管理する遷移モデルをそのまま送信するのではなく，割り当てられたセンサに対応する次元のみを切り出したモデルを送信する．

必要な情報が登録されたら，エッジおよびゲートウェイでそれぞれ処理を開始する．各エッジはセンサ群から計測値を受け取り，3.1 節で述べたとおり遷移モデルを用いて欠損値を復元する．その後，登録された各問合せ情報に従い時間窓での集約結果をゲートウェイへ送信する．ゲートウェイでは各エッジから集約結果を受け取り，3.2 節で述べたとおり損失した集約結果を復元する．この際，窓幅や問合せの種類に応じて適切に遷移モデルを切り替える．

上述のとおり，提案システムは必要最小限な構成でありながら，耐障害性を効率的に保証できる．提案システムでは障害によるデータ損失の近似的復元により，障害によらずシステム全体として常に処理を継続できるため，システムの可用性や処理遅延を高水準に維持できる．また処理の途中結果の定期的なチェックポインティングやデバイス単位の多重化の必要もないため，耐障害性保証コストを抑制できる．

4.2　効率的なウォーターマークの制御手法

データストリーム処理ではタプルが生成された時間（イベントタイム：event time）に基づく処理が基本となるため，処理対象のデータが全て揃ったことを保証するウォーターマークを待って処理結果を出力する．しかし，ウォーターマークをデータソース側で発行する場合，あるデバイスの遅れがシステム全体に伝搬し遅延が増大する．また，システム側で生成する場合，入力の遅延と欠損とを区別できないことから閾値に基づくヒューリスティックな生成 [12] が一般的であり，やはり遅延が発生する．よって，従来のウォーターマーク制御手法では障害による処理遅延が避けられず，提案アプローチによる処理遅延の削減効果を十分に発揮できない．

この問題に対して本稿では，提案アプローチにおける誤差保証の仕組みを活かした効率的なウォーターマーク制御手法を提案する．欠損値を含むデータストリームに対する集約処理は本質的に不確実であるため，ユーザは集約結果に対し許容可能な処理精度のしきい値を持つ必要がある．そこで，ユーザが指定する処理精度を満たした時点で処理に必要なデータが揃ったとみなすことで，従来手法に比べより早期かつ正確にウォータマークを生成できる．

本稿では処理精度を定めるためのしきい値として**要求信頼度** δ および**許容誤差** ϵ を用いる．つまり，ある時間窓における各デバイスの集約値が下記の式を満たすことを要求する．

$$\forall X \in \boldsymbol{X}, P(Y_X \in [\mu_{Y_X} - \epsilon, \mu_{Y_X} + \epsilon]) \geq \delta \tag{12}$$

ただし，欠損の度合いによってはこの式を満たさない可能性がある点に注意する．

式 (12) によって要求される推定の信頼度を表したとき，推定値の出力をする上で時間窓全ての観測値を必要としない場合があると分かる．極端な例を挙げれば，要求信頼度が $\delta = 0.01$ のように極めて小さい場合，一切の観測値を用いることなく事前

確率分布のみで式 (12) が満たされる可能性もある．したがって提案アプローチでは，時間窓の処理途中であっても，式 (12) が満たされると確定した時点で集約結果を出力できる．

式 (12) の成立を効率的に判定するため，信頼度の大小判定を分散の大小判定へと帰着させる．式 (12) は各信頼度が要求信頼度 δ を満たすかを見るが，信頼度の計算にはガウス分布に対する範囲積分が必要となる．つまり，単純には各時刻各デバイスに対して範囲積分の計算が必要となり効率が悪い．ここで，ガウス分布において期待値を中心とした積分範囲を用いるとき，期待値が積分結果へ影響を与えない点 [13] に着目する．このとき，式 (12) は下記のように書き換えられる．

$$\forall X \in \boldsymbol{X}, P(Y_X \in [-\epsilon, \epsilon]) \geqq \delta \tag{13}$$

$$\leftrightarrow \forall X \in \boldsymbol{X}, \int_{-\epsilon}^{\epsilon} \mathcal{N}(y \mid 0, \sigma_{Y_X}) dy \geqq \delta \tag{14}$$

つまり，与えられた要求信頼度 δ と誤差上限 ϵ に対して下記の式を解くことで，要求された処理精度を満たすために必要な **分散のしきい値** σ_θ が求められる．

$$\int_{-\epsilon}^{\epsilon} \mathcal{N}(y \mid 0, \sigma_{Y_X}) dy = \delta \tag{15}$$

ガウス分布の期待値を中心とした範囲積分であるため，分散の値が小さいほど信頼度の値は大きくなる．つまり，下記の式と式 (12) は同値である．

$$\forall X \in \boldsymbol{X}, \sigma_{Y_X} \leqq \sigma_\theta \tag{16}$$

5 評価実験

本章では，提案システムの処理遅延および可用性を実験により評価する．

5.1 実　　装

提案システムの性能評価実験を行うにあたり，プロトタイプシステムを構築した．既存のデータストリーム処理エンジンは主にクラウド環境を想定したものであり，エッジ環境への拡張が難しいため，一から新しく処理システムを実装した．エッジでは欠損値の復元とデータ集約の 2 つの処理を，ゲートウェイでは欠損値の復元処理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は Kafka [14] などのメッセージキューが一般的であるが，本実験では各欠損シナリオに対応した障害を模倣する必要があるため，特定時刻のデータ削除などの細かな制御が容易な PostgreSQL を採用した．また実装には，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 を使用しており，提案手法において必要な行列演算は全て numpy を用いて計算している．

実験にあたりデータソース用のシミュレータも別途作成した．想定する欠損シナリオに従い各センサの運転状況を模倣し，対応するエッジへと計測値を送信する．ただし，使用する実験データと同じく 1 分間隔でデータを送信すると 1 回の実験が完了するまで時間がかかりすぎるため，本実験では実際のデータの計測時刻とは独立してセンサのデータ送信間隔 T を別途指定する．特に指定が無い限り，$T = 0.1$[s] とする．また，センサのデータ送信間隔に合わせてエッジやゲートウェイの処理間隔もそれぞれ適切に設定する．

5.2 実 験 設 定

本実験で使用するデータセット，および評価方法についてそれぞれ説明する．

5.2.1 使用データセット

実験データ作成のため環境センサとして 24 個の 2JCIE-BL [15] を，エッジデバイスとして 5 台の Raspberry Pi 3 をそれぞれ使用し，研究室内の気温を 1 分間隔で 24 日間計測した．ただし，3 台のエッジにはセンサが 4 個ずつ，残り 2 台のエッジにはデバイスが 6 個ずつそれぞれ割り当てられているものとする．得られたデータの内，16 日分をモデル学習用，残りの 8 日分をテスト用に使用する．

本実験では以下の 2 種類の欠損シナリオを想定して，テストデータに対して人工的にデータ欠損を付与させる．

- **random シナリオ**：デバイス間の一時的な通信障害を模倣したシナリオであり，各デバイスの出力を欠損率 r でランダムに削除する．特に断りがない限り，$r = 0.1$ とする．
- **fault シナリオ**：センサやエッジの故障を模倣したシナリオであり，各デバイスの運転状況の遷移をシミュレートし，故障状態のときの出力データを削除する．ただしパラメータとして，運転状態から故障状態の遷移確率 r_f，および故障状態から運転状態の遷移確率 r_a を与えるものとする．特に断りがない限り，それぞれ $r_f = 0.1$, $r_a = 0.9$ とする．また，各エッジが故障状態へ遷移した際には主記憶上の処理の途中結果を削除し，故障状態から運転状態へ遷移した際にはキュー上に残っているデータのみを用いて処理を開始する．

5.2.2 評 価 方 法

本実験では，データ生成時刻に対するデータ出力時刻および応答時間によって提案システム性能を評価する．特に，処理遅延の評価では応答時間の平均値に，可用性の評価では応答時間の時間変化や分布にそれぞれ注目して評価する．[1] ここで，データ生成時刻とは各時間窓の始端となるデータが生成された時刻 t_i，データ出力時刻とは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 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．ただし，異なる時刻に実施した複数の実験結果を比較評価するため，各実験におけるデータ生成時刻の最小値に対する相対値を用いる．また，本実験では窓幅 w ステップ，滑り幅 l ステップの平均値問合せを対象とする．特に指定が無い限り，それぞれ $w = 10$, $l = 2$ とする．

以上の条件のもと，提案手法を含めた以下の 4 手法における処理結果を比較評価する．

- **ideal**：一切障害が発生しない理想的な状況下での処理

1：一般的には可用性の評価指標として，動作可能時間 T_a および動作不能時間 T_f に対して $T_a/(T_a + T_f)$ で求められる稼働率が用いられるが，多数のデバイスを用いた連続問合せ処理において T_a と T_f を正確に計測するのは困難であるため本実験では使用しない．

表 1　実験用サーバの構成

機器名	Dell PowerEdge R640
OS	Ubuntu 18.04.3 LTS
CPU	Intel(R) Xeon(R) Gold 6262V CPU @ 1.90GHz
メモリ	256GB
ストレージ	480GB

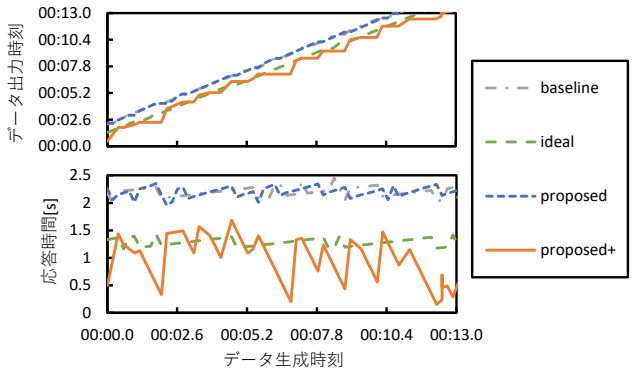


図 3　real シナリオにおけるシステムの処理状況

結果.

- **baseline**：at-most-once セマンティクスを採用した場合の処理結果．欠損値を近似的に復元せず各エッジで単純に集約処理を行い，ウォーターマークをヒューリスティックに制御する．すなわち，得られた最新時刻のデータの n ステップ前の時刻をウォーターマークとみなし，それ以前のデータは全て揃っていなくても集約する．特に断りが無い限り，$n = 5$ とする．
- **proposed**：提案システムの処理結果．ただし，ウォーターマークはヒューリスティックに制御する．
- **proposed+**：提案システムにおいて，要求信頼度 δ および誤差上限 ϵ を満たした時点でウォーターマークを発行する場合の処理結果．ただし，障害状況によっては時間窓終端までデータを待っても誤差要求を満たせない可能性もあるため，ヒューリスティックなウォーターマークと併用する．また，特に断りが無い限り，それぞれ $\delta = 0.9$, $\epsilon = 1.0$ とする．

また，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する．実験に使用したマシンの構成を表 1 に示す．Raspberry Pi などのエッジマシンの実機を用いた性能評価は今後の課題とする．

5.3　実 験 結 果

5.3.1　データ出力時刻および応答時間の時間変化

データ生成時刻に対するデータ出力時刻および応答時間の時間変化を評価する．各欠損シナリオにおける実験結果を図 3, 4 にそれぞれ示す．なお，細かな変化が分かるようデータ処理開始時点からおよそ 60 ステップ分の処理結果のみプロットしているが，これ以降の時刻においても同様の結果が得られた．

実験結果より，proposed は耐障害性保証のためにデータのバックアップや再処理を一切行わない baseline とほぼ同等の処理性能を有していることが見て取れる．また，proposed の応答時間が baseline のものに比べて一時的に著しく大きくなっていることもないため，提案システムは可用性を損なうことなく処

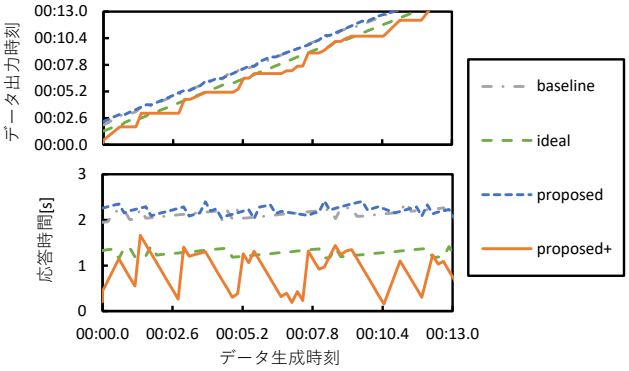


図 4　random シナリオにおけるシステムの処理状況

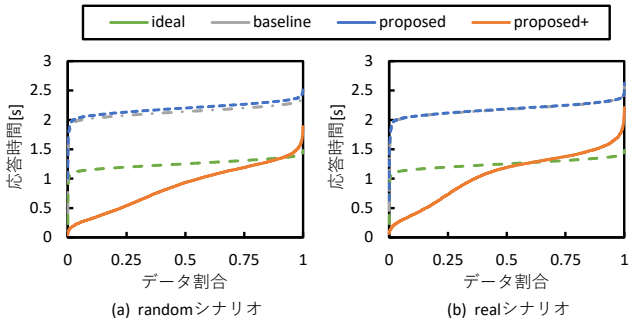


図 5　応答時間の分布

理を継続できていると分かる．本実験では各手法の処理精度は評価していないが，先行研究 [7] により提案手法は baseline に比べ処理精度を約 100 倍改善できると分かっている．以上より，提案システムでは処理性能を大きく損なうことなく高精度な集約結果を常に出力できるといえる．

proposed は ideal に比べ全体的に性能が劣っているが，これはウォーターマークをヒューリスティックに制御しているためであると考えられる．障害が一切発生しない ideal では窓幅だけ待てば時間窓中の全データが揃うため，応答時間はエッジやゲートウェイの処理間隔および窓幅 w に応じた値で安定している．しかし，各デバイスで障害が発生する proposed では更にウォーターマークの閾値 n ステップだけ余分に処理開始を待つ必要があるため，ideal の応答時間と差が生じている．

対して，proposed+の性能は ideal と同等以上となっており，本稿で提案した効率的なウォーターマーク制御手法の有効性が見て取れる．また，proposed+は常に baseline よりも早期な出力ができており，本実験でのパラメータ設定では時間窓中の全データが揃わずとも十分な精度で欠損値を復元できていると分かる．なお，他の手法に比べ応答時間の変動が大きくなっているが，これは各時刻で得られたデータ量に応じて適応的にウォーターマークを制御できて証拠であると考えらえる．

5.3.2　応答時間の分布

データ割合に対する応答時間の分布について評価する．各欠損シナリオでの実験結果を図 5 に示す．

この実験結果からも，proposed はどちらの欠損シナリオにおいても baseline と同等の性能を有しており，障害発生によらず常に処理を継続できていることが見て取れる．一方，proposed+はどちらの欠損シナリオにおいても ideal と同等以上の性能を

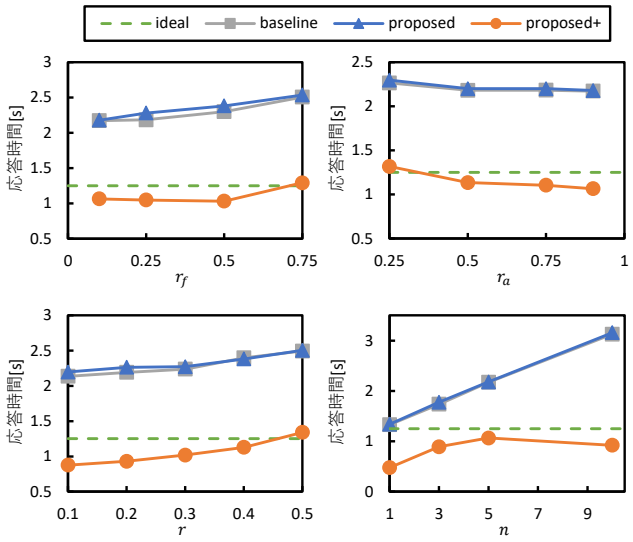


図 6　各パラメータに対する応答時間の変化

有しているものの，real シナリオでは random シナリオに比べて応答時間の立ち上がりが早くなっている．これは，各欠損シナリオでのデータ欠損の違いが原因であると考えられる．random シナリオでは全てのセンサで満遍なく欠損が発生するが，real シナリオでは故障状態から運転状態に遷移しない場合同じセンサでデータ欠損が連続してしまう．つまり同じ欠損率であっても各時間窓内での欠損状況をみたとき，random シナリオの方が特定のセンサに欠損が集まりやすいため，誤差要求を満たすまで時間がかかってしまうと考えられる．なお，real シナリオにおいても半分以上の時間窓は ideal より早期に出力できており，本稿で提案したウォーターマークの制御手法は十分有効に機能しているといえる．

5.3.3　各パラメータに対する応答時間の変化

各実験パラメータを変化させたときの応答時間の変化を評価する．

まずはじめに，障害の発生確率を制御する r_f, r_a, r およびヒューリスティックなウォーターマークの閾値 n を変化させたときの実験結果を図 6 に示す．ただし，n に対する実験では real シナリオを採用した．また，これらパラメータを変化させても ideal の結果は変わらないため，参考値として規定値での処理結果を示している．

実験結果より，real シナリオの故障状態への遷移確率 r_f および random シナリオの障害発生率 r は大きくなるほど欠損も増えるため，これらパラメータに対して応答時間も増加傾向にあることが見て取れる．また proposed+に注目すると，応答時間の変化違いから各欠損シナリオの違いが分かる．random シナリオでは欠損が全センサで満遍なく発生するため，r に対して応答時間が比例して増加している．一方，real シナリオでは特定のセンサに欠損が集中しやすいため，$r_f = 0.1$ の時点で応答時間が比較的大きくなっているものの $r_f = 0.5$ までは処理精度に大きな影響はなくほぼ一定の値で推移している．しかし，$r_f = 0.75$ になると処理精度への影響を無視できず応答時間が増加している．real シナリオの運転状態への遷移確率 r_a につ

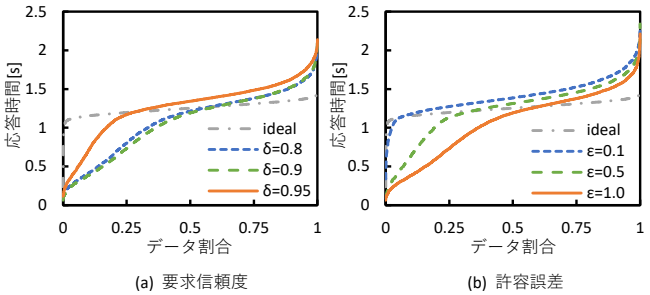


図 7　誤差要求に対する応答時間の分布の変化

いても，小さくなるほど欠損も増えるという違いはあるものの，r_f の場合と同様の結果となっている．

ヒューリスティックなウォーターマークの閾値 n は大きくなるほど処理を開始するまでの待ち時間が増加するため，proposed および baseline の応答時間は n に比例して増加している．proposed+でも $n = 5$ までは n の増加に伴い応答時間も増加しているが，proposed に比べて増加幅は緩やかである．これは，多くの時間窓において n ステップだけ待つよりも早くに誤差要求を満たした集約結果が得られているためだと考えられる．また，$n = 10$ では proposed+の応答時間が増加していない．これは，本実験設定においてほとんどの時間窓では 10 ステップ，すなわち窓幅全てのデータを待たずに誤差要求を満たした復元ができているためであると考えられる．以上より，本稿で提案したウォーターマーク制御手法を組み合わせることで，各デバイスからの出力が安定しないなどの n を大きく設定しなければならない状況下においても提案システムは処理遅延を大幅に抑制できるといえる．

次に，誤差要求に対する応答時間の変化を評価する．要求信頼度 δ および許容誤差 ϵ を変化させたときの応答時間の分布を図 7 に示す．ただし，proposed+以外はこれらパラメータの変化の影響を受けないため，参考値として規定値に対する ideal の実験結果のみを示している．また，これらは全て real シナリオを採用したとき実験結果である．

要求信頼度 δ が大きくなるほど誤差要求が厳しくなるため，処理結果の出力も遅くなると考えられる．実際，$\delta = 0.95$ の応答時間は $\delta = 0.9$ のものに比べて立ち上がりが早くなっており，応答時間に対する誤差要求の影響が見て取れる．一方，$\delta = 0.8$ の応答時間の分布は $\delta = 0.9$ のものとほぼ同じになっている．これは，本実験設定においてある程度のデータが揃った時点で $\delta = 0.9$ を満たす精度の処理結果を常に得られるため，それ以下の要求信頼度を指定しても影響が出なかったものと考えられる．また，許容誤差 ϵ についても誤差要求が厳しくなるほど応答時間の立ち上がりが早くなっており，要求信頼度と同様の傾向が見受けられる．以上の実験結果より，本稿で提案したウォーターマーク制御手法を用いることで，ユーザの誤差要求に対して適切なタイミングで処理結果を出力できるといえる．

6　おわりに

本稿では，耐障害性を近似的に保証するストリーム処理シス

テムの構築および評価実験について述べた．エッジ環境における近似的な耐障害性保証アプローチにおいて今まで曖昧であったシステム詳細を設計し，提案アプローチにおける誤差保証の仕組みを活用した効率的なウォータマークの制御方法を提案した．また，提案システムは様々な欠損シナリオにおいて，処理遅延や可用性を損なうことなく近似的に耐障害性を保証できることを実験により確認した．本実験により，本稿で提案したウォーターマーク制御手法を適用することで，提案システムは障害が一切発生しない状況と同等以上の処理能力を有すると分かった．

今後の課題として，エッジマシンの実機を使用したより現実的な環境での性能評価が挙げられる．特に，各デバイス間のネットワーク越しのデータ通信の影響の確認は必要であると考えており，現在実験に向けプロトタイプシステムを拡張している．その他，ゲートウェイの障害も考慮した手法の拡張や正規分布以外のモデルへの対応なども考えられる．

謝　　辞

本研究は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）から得られた結果による．また，本研究は JSPS 科研費（16H01722，20K19804）の助成，および JST 次世代研究者挑戦的研究プログラム（JPMJSP2125）の財政支援を受けたものである．この場を借りて「東海国立大学機構融合フロンティア次世代研究事業」に御礼申し上げる．

文　　献

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

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

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

[4] J.-H. Hwang, M. Balazinska, A. Rasin, U. Çetintemel, M. Stonebraker, and S. Zdonik, “High-availability algorithms for distributed stream processing,” in *Proc. ICDE*, pp. 779–790, 2005.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, “Discretized streams: A fault-tolerant model for scalable stream processing,” tech. rep., California Univ Berkeley Dept of Electrical Engineering and Computer Science, 2012.

[10] 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.

[11] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[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] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, “Model-based approximate querying in sensor networks,” *The VLDB Journal*, vol. 14, no. 4, pp. 417–443, 2005.

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

[15] “2JCIE-BL | 環境センサ.” https://components.omron.com/jp-ja/products/sensors/2JCIE-BL (Accessed: Jan. 8, 2022).