DEIM2020 H3-4(day1 p53)

チェックポインティングを考慮した近似的耐障害性保証

高尾　大樹† 　杉浦　健人† 　石川　佳治†

† 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,sugiura}@db.is.i.nagoya-u.ac.jp, ishikawa@i.nagoya-u.ac.jp

あらまし　分散並列ストリーム処理に注目が集まる一方，頑健な耐障害性保証によってシステム性能に悪影響が生じる例が報告されている．この問題に対し，近似的に耐障害性を保証することで処理性能を改善する手法が提案されたが，処理結果の最終的な誤差に対するユーザ要求の指定が難しいという課題がある．そこで，本稿では信頼度及び復帰時間に対する要求を直感的に指定可能な，近似的に耐障害性を保証する手法を提案する．本手法では，確率モデルに基づき内部状態を復元することで，処理の最終結果に対する直感的な誤差保証を提供する．また，チェックポインティングにより復帰時間の上限を保証するだけでなく，ユーザ要求の保証に必要な再処理データ量及びチェックポイント取得回数を最小化することでスループットの向上を図る．

キーワード　データストリーム処理，耐障害性，チェックポイント，近似処理

1 はじめに

リアルタイムに生成される大量のデータに基づいた意思決定支援のため，データストリーム処理に注目が集まっている．ストリーム処理によって断続的に生成されるデータをリアルタイムに処理することで，スマートビルディングやスマート工業などの新しい形のサービスを提供できる．これらのアプリケーションにおいてはリアルタイムな出力が常に求められるため，耐障害性保証はデータストリーム処理システムにおける重要な要件の一つである．

多くのシステムではチェックポインティングによる内部状態のバックアップと入力データの再送・再処理とに基づく復旧により，頑健に耐障害性を保証している．データが全てストレージにあることを想定する Hadoop [1] のようなバッチ処理システムとは異なり，ストリーム処理システムでは断続的に入力されるデータを効率的かつ安全に処理するためのアーキテクチャが必要となる．入力データや処理の途中結果などを主記憶上で管理することで高速なデータ処理が実現できるが，一方で障害によるデータ損失の影響が大きく，障害の発生によらず適切な出力を得る仕組みが必要である．そこで，Flink [2] や Samza [3] など多くの OSS（Open Source Software）ミドルウェアでは，チェックポイント，つまりローカルないしリモートのストレージに定期的に書き出した処理の途中結果を読み込み，それ以降の全入力データを再処理することで頑健な耐障害性保証を実現している．

しかし，頑健な耐障害性保証のための処理が目的の解析処理の性能を悪化させてしまうという問題がある．実際，Huang らはこのチェックポインティングによって Spark Streaming [4] のスループットが最大で 50%ほど低下してしまうことを示した [5]. 2025 年には世界中のデータ量が 175ZB に達し，リアルタイムデータはその 30%を占めるとの見立てもある [6] ことから，近い将来この問題点はより大きな影響を及ぼす恐れがある．特に，センサ数の爆発的な増加が予想される [7] ことからエッジコンピューティングの活用の機運が高まっているが，各エッジサーバで頑健に耐障害性を保証すると非常に大きなコストがかかると予想される．一方で，センシングデータには計測誤差が含まれ，またセンサ間の相関関係を考慮すれば一部のデータから他のデータを推定可能である [8] など，アプリケーションによっては頑健に耐障害性を保証するメリットは比較的小さいと考えられる．

この問題に対し，Huang らは近似的に耐障害性を保証することで処理性能を改善する手法を提案した [5] が，ユーザが求める SLA（Service Level Agreement）の直感的な指定が難しいという課題がある．この手法では，現在の内部状態（入出力キュー，処理の途中結果）とチェックポイントとして取得した過去の内部状態との変量が閾値を超えた場合にのみ新たなチェックポイントを取得することで，バックアップするデータ量及び頻度を削減しスループットを向上している．Huang らは耐障害性の近似を内部状態の変量の閾値によって定めているが，各閾値が処理の最終結果に与える影響の予測は難しい．例えば，入力キューの閾値は障害発生時の消失を許容する未処理のタプル数を示すが，もし消失したタプルの中に処理結果に大きな影響を与えるものが含まれる場合，最終的な処理結果はユーザが許容できない程に異なる可能性がある．また,「障害検知後 10 分以内に障害復帰を完了」などの障害復帰時間に対する SLA の保証は処理システムの実運用のうえで重要であるが，この手法には復帰時間の上限などを保証する枠組みがない．システム構成や処理内容などを考慮しながら要求を満たす閾値を実験的に指定するなどの対処も考えられるが，誤差保証との兼ね合いで自由度は低く，またパラメータ指定のコストも大きい．

そこで，本稿では信頼度及び復帰時間に対する要求を直感的に指定可能な，近似的に耐障害性を保証する手法を提案する．本手法では，確率モデルに基づき内部状態を復元することで，処理の最終結果に対する直感的な誤差保証を提供する．また，チェックポインティングにより復帰時間の上限を保証するだけ

sensor	time step							
	1	2	3	4	5	6	7	···
X_1	21	22	22	22	23	23	23	
X_2	21	21	22	22	22	23	23	···
X_3	19	20	20	20	19	20	21	

図 1　センサストリーム

でなく，ユーザ要求の保証に必要な再処理データ量及びチェックポイント取得回数を最小化することでスループットの向上を図る．

例として，図 1 のようなセンサストリームを考える．センサストリームは 3 つのセンサを持つが，このうち X_1 と X_2 には強い相関が，X_3 は他の二つセンサとの弱い相関がそれぞれ見られ，これらの相関関係は多次元ガウス分布によりモデル化できる．このとき，確率モデルを用いた推定により，一部のデータの処理結果から他のデータの処理結果を近似的に復元できる．今回の例では，$\{X_1, X_3\}$ の処理結果から X_2 の処理結果を推定できるものとする．

本手法ではこのようなセンサストリームに対して，信頼度要求としてユーザが指定する許容誤差及び許容誤り率を満たす範囲で再処理データを削減することで，障害復帰処理のコスト削減を図る．再処理データ量が 6 以下のときユーザの復帰時間要求を満たせるものとすると，復帰処理において全入力データを再処理する場合，復帰時間要求を満たすためには 2 単位時刻毎にチェックポイント取得の必要がある．一方，本手法では $\{X_1, X_3\}$ の再処理結果から X_2 の処理結果を推定することで，3 単位時刻毎にチェックポイントを取得すれば復帰時間要求を保証できる．すなわち，再処理対象の削減により障害復帰時間を短縮でき，それだけチェックポイントの取得間隔を拡大できる．これにより，外部 DB などとの通信オーバヘッドを削減でき，これがスループットの向上につながる．

また，チェックポイントから処理の途中結果を完全に復元することで最終的な誤差を抑えられるため，チェックポイント取得以降の再処理対象を更に削減できる．例えば，時刻 $t = 3$ でチェックポイントを取得した場合，以降は X_1 の処理結果から $\{X_2, X_3\}$ の処理結果を復元できる，といった具合である．このとき，6 単位時刻毎のチェックポイント取得で復帰時間要求を保証できるので，更なるスループットの向上が期待できる．

本稿の貢献は以下の通りである．

直感的な SLA の保証　本手法では，ユーザ要求として信頼度要求及び復帰時間要求を想定し，ユーザが直感的に理解可能な SLA の保証を提供する．障害発生時には確率モデルに基づく処理結果の推定により，許容誤差，及び許容誤り率を満たして内部状態を近似的に復元する．また，チェックポイント間隔に対する障害復帰時間を評価し，適切なチェックポイント間隔を設定することで障害復帰時間の上限を保証する．

チェックポイントを活用した効率的な耐障害性保証　チェックポインティングを効率的に取り入れることで，復帰時間要求を保証するだけでなく，障害復帰処理時の再処理データ量を削減する．これにより復帰時間要求を満たすのに必要なチェックポインティング間隔を緩和することで，外部 DB との通信オーバヘッドを削減し，SLA を満たす範囲でスループットを最大化する．

本稿の構成は以下の通りである．まず初めに，提案手法の概要を 2 章で述べる．次に，本手法で用いる確率モデルによる処理結果の推定について 3 章で述べる．また，スループット最大化のためのアプローチとして，再処理データの削減及びチェックポイント間隔の緩和についてそれぞれ 4 章及び 5 章で述べる．6 章では提案手法の評価実験について述べ，7 章では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について 8 章で述べる．

2　手法概要

本章では，提案手法による近似的耐障害性保証の概要を述べる．本稿では，同じ時刻を持ち，定期的にセンシングデータを測定する全 n 個のセンサ $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$ を扱う．つまり，システムに入力されるデータストリーム DS を各時刻 t におけるセンサデータ $\boldsymbol{x}^t = \{x_1^t, x_2^t, \ldots, x_n^t\}$ の無限の系列とする．

$$DS = \langle \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^t, \ldots \rangle \tag{1}$$

ただし本稿では，センサ属性や確率変数を大文字（X_1）で，センサデータを小文字（x_1）でそれぞれ表記するものとする．

合計及び平均の 2 つの集約問合せを処理対象とし，ある属性 $X_i \in \boldsymbol{X}$ に対する集約結果 Y_i の近似的な障害復帰を目指す．ただし，集約問合せは各時間窓 $\boldsymbol{x}^{(t,t+w]} \in W_{DS}$ に適用されるものとする．本稿では，窓幅を w とするとき，時間窓 $\boldsymbol{x}^{(t,t+w]}$ を時間幅 $(t, t+w]$ におけるセンサデータの有限の系列とする．

$$\boldsymbol{x}^{(t,t+w]} = \langle \boldsymbol{x}^{t+1}, \boldsymbol{x}^{t+1}, \ldots, \boldsymbol{x}^{t+w} \rangle \tag{2}$$

このとき，データストリーム DS は時間窓により以下の通り分割される．

$$W_{DS} = \left\{ \boldsymbol{x}^{(t,t+w]} \mid t = i \cdot w, i \in \mathcal{N}^0 \right\} \tag{3}$$

本手法では，確率モデルに基づく内部状態の復元により，ユーザが指定する信頼度要求を保証する．本稿では，信頼度要求として許容誤差 ϵ 及び許容誤り率 δ を想定する．すなわち，各時間窓 $\boldsymbol{x}^{(t,t+w]} \in W_{DS}$ の処理中に障害が発生した際には，各属性 $X_i \in \boldsymbol{X}$ の集約値 Y_i として

$$P(Y_i \in [y_i - \epsilon, y_i + \epsilon]) > 1 - \delta \tag{4}$$

を満たす y_i を出力することを保証する．

本手法では復帰時間要求を保証するため，内部状態，すなわち処理の途中結果のチェックポイントを活用して時間窓 $\boldsymbol{x}^{(t,t+w]}$ の集約結果を復元する．ただし，本稿では復帰時間要求として復帰時間の上限 T_{SLA} を想定する．本手法において想定するシステム構成，及び本手法による障害復帰の流れを図 2 に示す．時刻 t_{err} において障害が発生したとき，最新のチェックポイント $\boldsymbol{c}^k$ 及び推定に用いる確率モデルを DB からリストアする．た

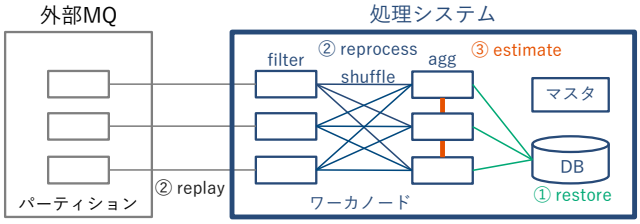

図 2　提案手法での障害復帰処理の流れ

だし，本稿ではチェックポイントの一貫した表記のため，時間窓 $\boldsymbol{x}^{(t,t+w]}$ の処理開始時点を $\boldsymbol{c}^0$（すなわち，$\tau_0 = t$）とし，$\boldsymbol{c}^k$ のチェックポイント間隔を $(\tau_{k-1}, \tau_k]$ と表すものとする．そして，次のチェックポイント $\boldsymbol{c}^{k+1}$ の取得時刻 τ_{k+1} までの一部のセンサ $\boldsymbol{O} \subseteq \boldsymbol{X}$ のデータ $\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}$ を再送・再処理する．本研究ではメッセジングキューとして Kafka [9] のような既存のメッセージシステムを想定し，再送と再処理はパイプライン的に並列処理するものとする．最後に，$\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}$ の再処理結果から残りのセンサの再処理結果を推定し，$\boldsymbol{c}^k$ と結合することで損失した内部状態のチェックポイント $\boldsymbol{c}^{k+1}$ を近似的に復元する．ここで，チェックポイント及び確率モデルのリストアと再処理結果の推定処理は，再処理データ量には依存せず一定の時間で処理が完了すると考えられる．一方で，$\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}$ の再送・再処理では処理時間が再処理データ量に大きく依存すると考えられる．よって，適切なタイミングでチェックポイントを取得し再処理データ量を一定量に抑えることで，障害復帰時間の上限 T_{SLA} を保証できる．チェックポイント取得時刻の具体的な決定方法については 5 章で詳しく述べる．

図 1 を用いて障害復帰時のチェックポイント取得の流れを説明する．時間窓 $\boldsymbol{x}^{(0,7]}$ に対して $\tau_1 = 3$ でチェックポイント $\boldsymbol{c}^1$ を取得し，時刻 $t = 5$ で障害が発生し処理の途中結果が失われたとする．処理システムは障害が発生した正確な時刻を知る術は無いため，次のチェックポイント $\boldsymbol{c}^2$ の取得時刻 $\tau_2 = 7$ までの X_1 のデータ $\boldsymbol{x}_1^{(3,7]}$ を再処理する．そして，$\boldsymbol{x}_2^{(3,7]}$ 及び $\boldsymbol{x}_3^{(3,7]}$ の処理結果を確率モデルに基づいて推定し，推定した値と $\boldsymbol{c}^1$ とを結合することで $\boldsymbol{c}^2$ を近似的に復元する．

また，本手法ではチェックポインティングを効率的に取り入れることで，ユーザ要求を満たしたスループットの最大化を実現する．チェックポイントのリストアにより部分的な処理結果を完全に復元することで，最終的な処理結果に対する誤差を抑制できる．よってチェックポイントの進行，つまり処理済みデータの増加に合わせて，信頼度要求を満たす範囲で再処理対象のセンサを削減できる．ここで，再処理対象のセンサ集合を $\boldsymbol{O}$，再送する時間幅（チェックポイント間隔）を l とした場合，再処理データ量は $|\boldsymbol{O}|l$ と表せる．すなわち，再処理対象のセンサ $\boldsymbol{O}$ を減らしただけチェックポイント間隔 l を増加させても，再処理データ量は一定に保てる．このように復帰時間要求を満たすのに必要なチェックポインティング間隔を緩和することで，外部 DB との通信オーバヘッドを削減し，SLA を満たす範囲でスループットを最大化する．

3　確率モデルに基づく内部状態の復元

本章では，本手法において基礎となる確率モデルに基づく内部状態の復元について述べる．確率モデルに基づくセンサ値の推定について 3.1 節で述べたあと，平均値及び合計値問合せへの拡張について 3.2 節で述べる．

3.1　確率モデルに基づく特定のセンサ値の推定

本研究ではセンサ $\boldsymbol{X}$ 間の相関関係を表現した多次元ガウス分布 $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma)$ を活用しセンサ値の推定を行う．なお，$\boldsymbol{\mu}$ 及び Σ はそれぞれ $\boldsymbol{X}$ の平均ベクトルと分散共分散行列を表す．

センサ集合 $\boldsymbol{X}$ が多次元ガウス分布に従うとき，範囲積分によってセンサ値を確率的に推定できる [8]．例えば，センサ $Y \in \boldsymbol{X}$ の値がある値 y である確率は，推定における許容誤差 ϵ を与えることで以下のように求められる．

$$P(Y \in [y-\epsilon, y+\epsilon]) = \int_{y-\epsilon}^{y+\epsilon} \mathcal{N}(y \mid \mu_y, \sigma_y) dy \tag{5}$$

つまり，ユーザから許容誤差 ϵ と許容誤り率 δ を受け取ることで，推定した値 $Y = y$ が信頼に足るか（確率が $1-\delta$ より大きくなるか）判断できる．

更に，事後確率分布を用いることで推定の精度を向上できる．いくつかのセンサ $\boldsymbol{O} \subseteq \boldsymbol{X}$ に対してその測定値 $\boldsymbol{o}$ が得られたとき，センサ $Y \in \boldsymbol{X} \setminus \boldsymbol{O}$ の事後確率を表すガウス分布 $\mathcal{N}(y \mid \mu_{Y|\boldsymbol{o}}, \sigma_{Y|\boldsymbol{o}})$ は以下の式から求められる [8]．

$$\mu_{Y|\boldsymbol{o}} = \mu_Y + \Sigma_{Y\boldsymbol{O}}\Sigma_{\boldsymbol{O}\boldsymbol{O}}^{-1}(\boldsymbol{o} - \boldsymbol{\mu}_{\boldsymbol{O}}) \tag{6}$$

$$\sigma_{Y|\boldsymbol{o}} = \Sigma_{YY} - \Sigma_{Y\boldsymbol{O}}\Sigma_{\boldsymbol{O}\boldsymbol{O}}^{-1}\Sigma_{\boldsymbol{O}Y} \tag{7}$$

ここで，各記号の添字はベクトルないし行列から添え字に対応する次元を抽出したことを示す．例えば，μ_Y は Y に対応する次元の値を 平均ベクトル $\boldsymbol{\mu}$ から抽出したもの（つまりセンサ Y の期待値）であり，$\Sigma_{Y\boldsymbol{O}}$ は 分散共分散行列 Σ の Y 行 から $\boldsymbol{O}$ に対応する列を抽出したもの（つまりセンサ Y に対する参照センサ集合 $\boldsymbol{O}$ の共分散ベクトル）である．事後確率分布の分散は事前分布のそれよりも減少するため，より高精度な推定が可能となる．

また，式 (7) に示す事後確率分布の分散は参照センサ集合 $\boldsymbol{O}$ の実測値 $\boldsymbol{o}$ に依存しない [8] ことから，以後 $\sigma_{Y|\boldsymbol{O}}$ と表記する．

3.2　平均値及び合計値問合せへの拡張

本手法では，各入力データがガウス分布に従うとき平均値及び合計値問合せの出力値もガウス分布に従うことから，出力値に対するガウス分布を用いて損失した処理結果を推定する．具体的には，最新のチェックポイント $\boldsymbol{c}^k$ からその次のチェックポイント $\boldsymbol{c}^{k+1}$ 取得までの，再処理対象のセンサ $\boldsymbol{O} \subseteq \boldsymbol{X}$ のデータ $\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}$ の再処理結果から $\boldsymbol{x}^{(\tau_k,\tau_{k+1}]}$ の処理結果を直接推定する．以降では，平均値問合せを想定して推定方法について議論を進めるが，合計値に対しても同様の議論が可能である．

時間的な相関を考慮しない場合，あるセンサ $X_i \in \boldsymbol{X}$ の時間幅 $(\tau_k, \tau_{k+1}]$ における平均値 Y_i^{k+1} 及びそのガウス分布の期待

値・分散はそれぞれ次の通りである．

$$Y_i^{k+1} = \frac{\Sigma_{t\in(\tau_k,\tau_{k+1}]} X_i^t}{\tau_{k+1}-\tau_k} \tag{8}$$

$$\mu_{Y_i^{k+1}} = \frac{\Sigma_{t\in(\tau_k,\tau_{k+1}]} \mu_{X_i^t}}{\tau_{k+1}-\tau_k} \tag{9}$$

$$\sigma_{Y_i^{k+1}} = \frac{\Sigma_{t\in(\tau_k,\tau_{k+1}]} \sigma_{X_i^t}}{(\tau_{k+1}-\tau_k)^2} \tag{10}$$

つまり，時間幅 $(\tau_k, \tau_{k+1}]$ におけるセンサ X_i の平均値 Y_i^{k+1} の期待値と分散は，各時刻におけるセンサ X_i の期待値及び分散の線形和でそれぞれ計算可能である．

また，センサ $X_i, X_j \in \boldsymbol{X}$ の平均値 Y_i^{k+1}, Y_j^{k+1} 間の共分散も以下の通り，各時刻におけるセンサ X_i, X_j の共分散の線形和で計算可能である．

$$\sigma_{Y_i^{k+1}Y_j^{k+1}} = \frac{\Sigma_{t\in(\tau_k,\tau_{k+1}]} \sigma_{X_i^t X_j^t}}{(t_{k+1}-t_k)^2} \tag{11}$$

以上より，時間幅 $(\tau_k, \tau_{k+1}]$ における集約値のガウス分布は各時刻のガウス分布から計算可能であるとわかる．集約値のガウス分布を計算できれば，3.1 節で述べた通りの方法で，部分ストリーム $\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}$ の再処理結果から $\boldsymbol{x}^{(\tau_k,\tau_{k+1}]}$ の処理結果を推定できる．

4 チェックポイントを考慮した再処理データの削減

本手法ではチェックポイントを効率的に取り入れることで，復帰時間要求を保証するだけでなく，再処理対象のセンサを削減し復帰処理コストを抑制する．ただし，再処理対象のセンサを削減しても信頼度要求をかわらず保証する必要があるため，再処理対象のセンサに対する推定信頼度の適切な評価，及び信頼度要求を満たした安全な再処理対象センサの削減が重要となる．

そこで本章では，信頼度要求を満たした再処理対象センサの最小化問題を定義し，信頼度要求を理論的に保証可能な再処理対象センサの決定方法を述べる．

4.1 再処理対象センサの最小化問題

信頼度要求を満たした再処理対象センサ $\boldsymbol{O} \subseteq \boldsymbol{X}$ の最小化問題を定義するため，まずは再処理対象のセンサに対する復元の信頼度を定義する．時間窓 $\boldsymbol{x}^{(t',t'+w]}$ に対してチェックポイント $\boldsymbol{c}^k$ を取得できているとき，最悪の状況として，以降のチェックポイント $\boldsymbol{c}^j$ $(j>k)$ の取得間隔 $(\tau_{j-1}, \tau_j]$ すべてにおいて障害が発生することが考えられる．このとき，チェックポイント $\boldsymbol{c}^k$ と時間幅 $(\tau_k, t'+w]$ におけるセンサ $\boldsymbol{O}$ のデータ $\boldsymbol{o}^{(\tau_k,t'+w]}$ の再処理結果から $\boldsymbol{x}^{(t',t'+w]}$ の集約値を推定することと同義になる．従って，$\boldsymbol{c}^k$ と $\boldsymbol{o}^{(\tau_k,t'+w]}$ から復元した全ての集約結果がユーザ指定の信頼度要求である許容誤差 ϵ 及び許容誤り率 δ を満たすことを保証できればよい．

本稿ではこれを評価するため，$\boldsymbol{c}^k$ と $\boldsymbol{o}^{(\tau_k,t'+w]}$ から推定した各属性の信頼度（推定の確率）の最小値を用いて時間窓全体の復元の信頼度を定める．

定義 1. 許容誤差 ϵ，時間窓 $\boldsymbol{x}^{(t',t'+w]}$ に対する最新のチェックポイント $\boldsymbol{c}^k$，及び再処理対象のセンサ $\boldsymbol{O} \subseteq \boldsymbol{X}$ が与えられたとする．ある属性 $X_i \in \boldsymbol{X}$ に対する集約結果を確率変数 Y_i，システムによる復元値を y_i でそれぞれ表すとき，$\boldsymbol{c}^k$ 及び $\boldsymbol{O}$ による耐障害性保証の信頼度を以下の式で求める．

$$R(\boldsymbol{c}^k, \boldsymbol{O}) = \min_{X_i\in\boldsymbol{X}} P(Y_i \in [y_i-\epsilon, y_i+\epsilon] \mid \boldsymbol{c}^k, \boldsymbol{o}^{(\tau_k,t'+w]}) \tag{12}$$

■

すなわち，復元の信頼度 $R(\boldsymbol{c}^k, \boldsymbol{O})$ が $1-\delta$ より大きくなれば，提案手法により信頼度要求を満たした復元が可能である．

よって，最新のチェックポイント $\boldsymbol{c}^k$ に対する信頼度要求を満たした再処理対象センサの最小化問題は以下の通り定義できる．

定義 2. 許容誤差 ϵ，及び許容誤り率 δ が与えられたとき，時間窓 $\boldsymbol{x}^{(t',t'+w]}$ の最新のチェックポイント $\boldsymbol{c}^k$ に対して信頼度要求を満たした最小数の再処理対象センサ $\boldsymbol{O}$ は以下の式の通り表される．

$$\begin{aligned} &\text{minimize} && |\boldsymbol{O}| \\ &\text{subject to} && \boldsymbol{O} \subseteq \boldsymbol{X} \\ & && R(\boldsymbol{c}^k, \boldsymbol{O}) > 1-\delta \end{aligned} \tag{13}$$

■

4.2 信頼度要求を満たした再処理対象の削減

本節では，ユーザ指定の信頼度要求である許容誤差 ϵ 及び許容誤り率 δ を満たした最小数の再処理対象センサ $\boldsymbol{O}$ の効率的な決定方法について述べる．最小の $\boldsymbol{O}$ を愚直に求める場合 2^n 通りのセンサの組み合わせそれぞれに対して信頼度 $R(\boldsymbol{c}^k, \boldsymbol{O})$ を計算する必要があり，計算コストの観点から望ましくない．そこで，耐障害性の信頼度が事前計算可能である性質を活用して，信頼度要求を満たすのに必要な再処理区間の分散を理論的に導出し，目標の分散を満たす $\boldsymbol{O}$ を貪欲的に決定する．以降では，時間窓 $\boldsymbol{x}^{(t',t'+w]}$ での平均値問合せを想定して議論を進めるが，合計値問合せに対しても同様の議論が可能である．

まずはじめに，最新のチェックポイント $\boldsymbol{c}^k$ に対して再処理対象のセンサ $\boldsymbol{O} \subseteq \boldsymbol{X}$ が決まったとき，耐障害性の信頼度が事前計算可能であることを示す．あるセンサ $X_i \in \boldsymbol{X}$ の平均値 Y_i 及びその期待値と分散は，$\boldsymbol{c}^k$ における X_i の処理結果 c_i^k と最悪の場合の再処理区間 $(\tau_k, t'+w]$ の復元結果 Y_i^{rec} に対してそれぞれ以下の通りである．

$$Y_i = \frac{(\tau_k - t')c_i^k + (t'+w-\tau_k)Y_i^{rec}}{w} \tag{14}$$

$$\mu_{Y_i} = \frac{(\tau_k - t')\mu_{c_i^k} + (t'+w-\tau_k)\mu_{Y_i^{rec}}}{w} \tag{15}$$

$$\sigma_{Y_i} = \frac{(\tau_k - t')^2\sigma_{c_i^k} + (t'+w-\tau_k)^2\sigma_{Y_i^{rec}}}{w^2} \tag{16}$$

ここで重要となるのは，式 (7) にあるように，事後確率分布の分散 $\sigma_{Y|\boldsymbol{O}}$ が計測値に依存しない点である．つまり，再処理対象のセンサ $\boldsymbol{O}$ を決定したとき，各時刻の分散 $\sigma_{X_i^t|\boldsymbol{O}}$ の値は事前計算できる．また，式 (10) より各時刻の分散から再処理区間

の分散 $\sigma_{Y_i^{rec}}$ も計算できるので，σ_{Y_i} も事前計算可能である．これは，処理を開始する前にセンサ $\boldsymbol{O}$ で $R(\boldsymbol{c}^k, \boldsymbol{O}) > 1-\delta$ を満たせるかが確認可能であることを示す．

次に，信頼度要求を満たすのに必要な再処理区間の分散の理論的な導出について述べる．近似的に復元した平均値として Y_i の期待値 μ_{Y_i} を用いる場合，センサ X_i に関する信頼度は以下の式で計算できる．

$$\int_{\mu_{Y_i}-\epsilon}^{\mu_{Y_i}+\epsilon} \mathcal{N}(y_i \mid \mu_{Y_i}, \sigma_{Y_i}) dy_i \tag{17}$$

更に，$Y_i' = Y_i - \mu_{Y_i}$ と置くとき，この確率変数に対する平均は 0，分散は σ_{Y_i} となるため，式 (17) は以下の式により計算できる．

$$\int_{-\epsilon}^{\epsilon} \mathcal{N}(y_i' \mid 0, \sigma_{Y_i}) dy_i' > 1-\delta \tag{18}$$

ここで，この式を満たす最小の σ_{Y_i} を $\sigma^*_{Y_i}$ としたとき，式 (16) から以下の式が導ける．

$$\begin{aligned} &\forall X_i \in \boldsymbol{X}, \sigma_{Y_i^{rec}|\boldsymbol{O}} \leqq \frac{w^2\sigma^*_{Y_i} - (\tau_k - t')^2\sigma_{\boldsymbol{c}_i^k}}{(t'+w-\tau_k)^2} \\ &\rightarrow \sigma_{Y_i|\boldsymbol{O}^{(t',t'+w]}} \leqq \sigma^*_{Y_i} \end{aligned} \tag{19}$$

つまり，各 $\sigma_{Y_i^{rec}|\boldsymbol{O}}$ が式 (19) の前件を満たせば，復元した平均値はユーザが指定した信頼度要求を満たす．更に，信頼度要求を満たすための平均値の分散 $\sigma^*_{Y_i}$ はセンサ X_i に依存せず，$\forall X_i, X_j \in \boldsymbol{X}, \sigma^*_{Y_i} = \sigma^*_{Y_j}$ である．したがって，以下の式を満たす $\boldsymbol{O}$ を得られれば，時間窓中の全てのセンサにおける平均値を信頼度要求を満たして復元できる．

$$\forall X_i \in \boldsymbol{X}, \sigma_{\boldsymbol{c}_i^{rec}|\boldsymbol{O}} \leqq \frac{w^2\sigma^*_Y - (\tau_k - t')^2\sigma_{\boldsymbol{c}_i^k}}{(t'+w-\tau_k)^2} \tag{20}$$

ただし，σ^*_Y は以下の式を満たす最小の値であるとする．

$$\int_{-\epsilon}^{\epsilon} \mathcal{N}(y \mid 0, \sigma_Y) dy > 1-\delta \tag{21}$$

最後に，目標の分散を満たす $\boldsymbol{O}$ の貪欲的な決定方法を述べる．バックアップするセンサの組合せ $\boldsymbol{O} \subseteq \boldsymbol{X}$ に対して新しい要素 $X_i \in \boldsymbol{X} \backslash \boldsymbol{O}$ を加えると，復元結果のモデルの分散は減少する．分散が減少するほど推定の精度が向上するといえるので，$\boldsymbol{O}$ の選択の基本方針として，なるべく分散が減少するようなセンサを貪欲的に選んでいけばよい．

本稿では，再処理対象のセンサ集合 $\boldsymbol{O}$ に対する要素 X_i の追加による分散の減少を次式の $Var(\boldsymbol{O}, X_i)$ により評価する．

$$Var(\boldsymbol{O}, X_i) = \max_{X_j \in \boldsymbol{X}} \sigma_{Y_j^{rec}|\boldsymbol{O}} - \sigma_{Y_j^{rec}|\boldsymbol{O} \cup \{X_i\}} \tag{22}$$

このとき，$Var(\boldsymbol{O}, X_i)$ が一番大きくなるような $X_{new} \in \boldsymbol{X} \backslash \boldsymbol{O}$ を貪欲的に $\boldsymbol{O}$ に追加すればよい．

$$X_{new} = \arg\max_{X_i \in \boldsymbol{X} \backslash \boldsymbol{O}} Var(\boldsymbol{O}, X_i) \tag{23}$$

上述の議論を，貪欲的な再処理対象センサの決定アルゴリズムとして図 3 にまとめる．

Input: センサ集合 $\boldsymbol{X}$, 目標分散値 σ^*
Output: 再処理対象のセンサ集合 $\boldsymbol{O}$

```
O ← ∅
while Var(O) > σ* do
    X_new ← arg max_{X_i ∈ X\O} Var(O, X_i)
    O ← O ∪ {X_new}
```

図 3　貪欲法に基づく再処理対象センサ決定アルゴリズム

5　チェックポイント間隔の緩和

本手法では，復帰時間要求を満たす範囲でチェックポイント間隔を緩和することで，スループットの最大化を図る．そのためには，再処理データ量に対する復帰時間を適切に評価し，最適なチェックポイント取得計画を設定する必要がある．

そこで本章では，復帰時間要求を満たしたチェックポイント間隔の最大化問題を定義し，チェックポイント取得計画を事前に決定する方法を述べる．

5.1　チェックポイント間隔の最大化問題

本手法では図 2 に示す通り，時間窓 $\boldsymbol{x}^{(t',t'+w]}$ の最新のチェックポイント $\boldsymbol{c}^k$ に対して，以下の 3 ステップによって信頼度要求及び復帰時間要求を満たした障害復帰処理を実現する．

（1） 最新のチェックポイント $\boldsymbol{c}^k$ のリストア
（2） 部分ストリーム $\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}$ の再送・再処理
（3） 再処理結果から他のデータの処理結果の推定

各ステップにかかる時間をそれぞれ $T_{res}, T_{rep}, T_{est}$ とおくと，本手法の障害復帰時間 T_{rec} は以下で求められる．

$$T_{rec} = T_{res} + T_{rep} + T_{est} \tag{24}$$

ここで，T_{rep} は再処理データ量に依存し，T_{res} 及び T_{est} は再処理データ量に依存せず一定の時間となると考えられる．そこで本稿では，障害復帰時間 T_{rec} を再処理データ量 $|\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}|$ に対する関数 $T_{rec}(|\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}|)$ として扱う．なお，再処理データ量 $|\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}|$ は再処理対象のセンサ数 $|\boldsymbol{O}|$ と復帰処理を行う時間幅 $\tau_{k+1} - \tau_k$ の積で求められる．また，関数の詳細についてはシステム構成に依存するため，事前の測定によって設計する必要がある．

従って，最新のチェックポイント $\boldsymbol{c}^k$ に対する再処理センサ $\boldsymbol{O}$ が与えられたとき，復帰時間要求を満たしたチェックポイント間隔の最大化問題は以下の通り定義できる．

定義 3. 時間窓 $\boldsymbol{x}^{(t',t'+w]}$ の最新のチェックポイント $\boldsymbol{c}^k$ に対する再処理対象のセンサ $\boldsymbol{O}$ が与えられたとき，復帰時間要求 T_{SLA} に対して以下の式で表される時刻 τ_{k+1} においてチェックポイント $\boldsymbol{c}^{k+1}$ を取得する．

$$\begin{aligned} &\text{maximize} && \tau_{k+1} \\ &\text{subject to} && \tau_{k+1} \in (\tau_k, t'+w] \\ & && T_{rec}(|\boldsymbol{o}^{(\tau_k,\tau_{k+1}]}|) \leqq T_{SLA} \end{aligned} \tag{25}$$

■

Input: センサ集合 $\boldsymbol{X}$, 時間幅 $(t', t'+w]$, 最新のチェックポイント $\boldsymbol{c}^k$

Output: チェックポイント取得時刻系列 $\boldsymbol{\tau}$

1 $\tau_{max} \leftarrow \tau_k$
2 $\boldsymbol{\tau} \leftarrow \langle \tau_{max} \rangle$
3 **while** $\tau_{max} < t' + w$ **do**
4 $\quad \boldsymbol{O} \leftarrow$ 式 (13) により求めた再処理対象のセンサ集合
5 $\quad \tau_{max} \leftarrow$ 式 (25) により求めたチェックポイント取得時刻
6 $\quad \boldsymbol{\tau} \leftarrow \boldsymbol{\tau} + \langle \tau_{max} \rangle$

図 4　最適なチェックポイント取得時刻系列の導出アルゴリズム

この問題については，障害復帰時間を表す関数 $T_{rec}(|\boldsymbol{o}^{(\tau_k, \tau_{k+1}]}|)$ が次回チェックポイント取得時刻 τ_{k+1} に対して単調増加すると考えられるため，探索的アプローチにより効率的に解ける．

5.2　チェックポイント取得計画の決定

これまでの議論では，式 (25) を解くことで再処理対象のセンサ $\boldsymbol{O}$ に対して復帰時間要求を満たす最大のチェックポイント取得時刻 $\boldsymbol{c}^{k+1}$ を求められることを示した．しかし，実際の運用においてチェックポイント取得毎に次回のチェックポイント取得時刻をオンラインで求めるのは非効率的であるため，$\boldsymbol{x}^{(t', t'+w]}$ に対して SLA を保証しながらスループットを最大化する全チェックポイントの取得時刻系列 $\boldsymbol{\tau} = \langle \tau_1, \tau_2, \ldots, \tau_m \rangle$ を事前計算したい．

そこで本稿では，時間窓 $\boldsymbol{x}^{(t', t'+w]}$ に対するチェックポイント取得時刻系列 $\boldsymbol{\tau}$ を事前計算する方法を提案する．ここで，障害が発生せずチェックポイント $\boldsymbol{c}^{k+1}$ が取得できたと仮定すれば，再処理対象のセンサ $\boldsymbol{O}'$ も事前に決定できる点に注目する．すなわち，式 (13) に示す再処理データの最小化問題と式 (25) に示すチェックポイント間隔の最大化問題を交互に解くことで，時間窓 $\boldsymbol{x}^{(t', t'+w]}$ の処理中に一度も障害が発生しなかった場合の最適な $\boldsymbol{\tau}$ を求められる．

上述の議論を，最適なチェックポイント取得時刻系列の導出アルゴリズムとして図 4 にまとめる．ただし，チェックポイント取得時刻系列間の二項演算子 + を以下の通り定義する．

定義 4. チェックポイント取得時刻系列 $\boldsymbol{\tau}_a = \langle \tau_{a_1}, \tau_{a_2}, \ldots, \tau_{a_i} \rangle$ 及び $\boldsymbol{\tau}_b = \langle \tau_{b_1}, \tau_{b_2}, \ldots, \tau_{b_j} \rangle$ に対して，二項演算子 + を以下の通り定義する．

$$\boldsymbol{\tau}_a + \boldsymbol{\tau}_b = \langle \tau_{a_1}, \tau_{a_2}, \ldots, \tau_{a_i}, \tau_{b_1}, \tau_{b_2}, \ldots, \tau_{b_j} \rangle \qquad (26)$$

■

なお，ある時間幅 $(\tau_{j-1}, \tau_j]$ において障害が発生した場合には，復元した $\boldsymbol{c}^j$ を $\boldsymbol{c}^k$ として再計算することで，τ_j 以降の最適なチェックポイント取得時刻系列 $\boldsymbol{\tau}' = \langle \tau_{j+1}, \tau_{j+2}, \ldots, \tau_{m'} \rangle$ を求められる．

図 1 を用いて，時間窓 $\boldsymbol{x}^{(0,7]}$ に対する最適なチェックポイント取得時刻系列の導出の流れを説明する．アルゴリズムの入力はセンサ集合 $\boldsymbol{X} = \{X_1, X_2, X_3\}$，時間幅 $(0, 7]$，最新のチェックポイント $\boldsymbol{c}^k = \boldsymbol{c}^0$ $(\tau_0 = 0)$ となるため，$\tau_{max} = 0$，$\boldsymbol{\tau} = \langle 0 \rangle$ として初期化できる（図 4 中 1,2 行目）．チェックポイントを取得していないとき $\{X_1, X_3\}$ の再処理結果から $\{X_2\}$ の処理結果を推定できるとすると，式 (13) により $\boldsymbol{O} = \{X_1, X_3\}$ が得られる（図 4 中 4 行目）．再処理データ量が 6 以下のときユーザの復帰時間要求を満たせるものとすると，復帰処理において $\{X_1, X_3\}$ のデータの再処理が必要であるので式 (25) により $\tau_{max} = 3$ が得られる（図 4 中 5 行目）．$\tau_{max} < 7$ であるので，次に $\tau_1 = 3$ でチェックポイント $\boldsymbol{c}^1$ を取得したときの状況を考える．このとき $\{X_1\}$ の再処理結果から $\{X_2, X_3\}$ の処理結果を推定できるとすると，式 (13) から $\boldsymbol{O} = \{X_1\}$ が得られ，式 (25) から時間窓終端の $\tau_{max} = 7$ が得られる．以上より，$\boldsymbol{\tau} = \langle 0, 3, 7 \rangle$ が得られ，$\boldsymbol{x}^{(0,7]}$ の処理中 $\tau_1 = 3$ でチェックポイント $\boldsymbol{c}^1$ のみを取得すれば良いと分かる．

表 1　使用マシンスペック

OS	Ubuntu 14.04.1 LTS 64bit
CPU	Intel(R) Xeon(R) CPU E5620 @ 2.40GHz
メモリ	32GB

6　実　　験

本章では，提案手法の妥当性及び効率性について実験により評価する．提案手法の妥当性として信頼度要求を満たした復元が可能か，また全データを再処理するよりも高速な障害復帰が可能かについて検証する．また，提案手法の効率性として，復帰時間及びチェックポイント取得回数の削減能力について評価する．

6.1　復帰時間に関する評価実験

まず初めに，センサ数 n と再処理センサ数 $|\boldsymbol{O}|$ を変化させたときの，リストア時間 T_{res}，再処理時間 T_{rep}，推定時間 T_{est} をそれぞれ測定し，提案手法による復帰時間短縮の効果について確認する．ただし，本実験では表 1 に示すマシンを使用し，提案手法の障害復帰処理を Apache Flink [2] 上で疑似的に再現したプログラムを 1 スレッドで実行したときの処理時間によって評価する．また，再処理データとしてランダムに生成した系列数 10^5 の人工データを使用する．

実験結果を図 5 に示す．なお，リストア時間 T_{res} については常に 2[ms] 以下であったため省略している．各処理時間の変化についてみると，再処理時間 T_{rep} は再処理センサ数に比例して増加している一方，推定時間 T_{est} はセンサ数 n に依存して増加している．これは，推定処理において共分散行列を用いた計算が支配的となるためと考えられる．以上より，あるセンサ数（本実験では $n \leqq 10^3$）までは，提案手法は全てのセンサデータを再処理するよりも高速に障害復帰できると分かる．一方，それ以上のセンサ数（本実験では $n \geqq 10^4$）においては 1 スレッドだけでは高速な障害復帰ができないため，並列処理により 1 スレッドあたりのセンサ数を抑制する必要があるといえる．そのためにはセンサ間の相関関係を効率的に活用するキー割当などの工夫が必要であるが，これについては今後の課題である．

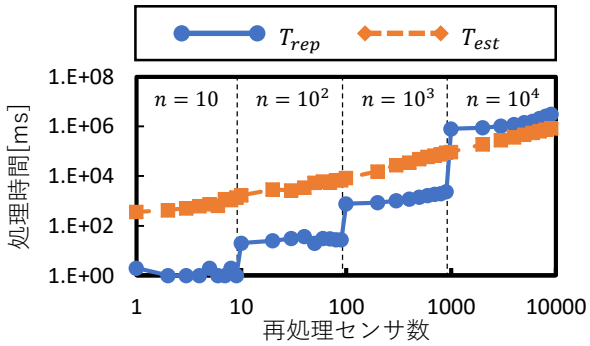

図 5　復帰処理の各ステップでの所用時間の変化

以降はセンサ数 $n = 100$ として実験を行うが，この時のあるチェックポイント c^k に対する復帰処理の各ステップの所用時間を今回の実験結果から次式の通り定める．

$$T_{res} + T_{est} = 28.33[ms] \tag{27}$$

$$T_{rep} = 9.332|o^{(\tau_k, \tau_{k+1}]}| \times 10^{-4}[ms] \tag{28}$$

6.2　チェックポイント取得回数に関する評価実験

次に，提案手法によるチェックポイント取得回数の削減効果について評価する．許容誤差 ϵ，許容誤り率 δ，復帰時間上限 T_{SLA} をそれぞれ変化させたときのチェックポイント取得回数を，図 4 に示すアルゴリズムに基づいて求め，以下の手法のものと比較評価する．

exactly-once　チェックポイント以降の全データを再処理

緩和なし　チェックポイントの取得状況を考慮せず，常に時間窓の開始時点 c^0 に対する再処理センサ $\boldsymbol{O}$ を用いて推定

ただし，本実験では対象問合せとして窓幅 $w = 10^4$，滑り幅 $l = 10^3$ の時間窓に対する平均問合せを想定し，人手で作成した部分的に相関関係のあるセンサデータを用いる．今回作成した実験データは 100 のセンサからなり，互いに相関関係をもつ 5 つのグループに分割できる．すなわち，各グループそれぞれ 20 のセンサが存在し，同グループ内のセンサデータから高精度な推定が可能である．ただし，各センサは単位時刻毎に定期的にデータを出力し，データの欠損は存在しないものとする．上述の実験データを作成するため，以下の条件のもと R 言語の rmvnorm 関数 [10] により 10 万タプルのデータ系列を生成した．

期待値　全センサに対して期待値を 0.0 に設定する．

分散・共分散　各センサに対して分散を 60 に設定する．また，同グループのセンサに対する共分散を 30，それ以外のセンサに対する共分散を 0 に設定する．

実験結果を図 6 に示す．現在既存システムにおいて一般的に用いられている exactly-once のアプローチでは，信頼度要求の変化に対して常に一定数のチェックポイントの取得が必要である．しかし，一部のセンサの処理結果から残りのセンサの処理結果を確率モデルによって推定することで，特に信頼度要求が緩い場合においてチェックポイント取得回数を削減できている．厳しい信頼度要求場合，単純に推定するだけでは良い結果が得られないが，提案手法はチェックポイント取得回数を削減できる．これは，チェックポイントの取得状況を考慮することで，処理が進むにつれ再処理センサ数を削減でき，これによりチェックポイント取得間隔を緩和できるためである．特に，復帰時間要求が厳しい場合においてチェックポイント取得回数の削減の効果が大きく出ていることから，リアルタイム性が要求されるようなアプリケーションにおいて提案手法は有効であるといえる．

6.3　推定精度に関する評価実験

最後に，提案手法による処理結果の復元成功率を評価する．本実験では，図 4 に示すアルゴリズムから求めた各チェックポイントに対して，提案手法により復元した処理結果と真値との差が許容誤差 ϵ の範囲に収まるときに復元に成功したとみなす．なお，対象クエリ及び実験データは 6.2 節と同じとする．

実験結果を図 7 に示す．パラメータの変化に対して正答率の上下が見られるが，これは主に再処理対象のセンサ数の減少による正答率の低下と信頼度要求の緩和による正答率の上昇によるものである．しかし，提案手法による復元成功率は常に信頼度要求以上であり，提案手法はユーザ要求を満たした処理結果の近似的復元が可能であるといえる．

7　関 連 研 究

本章では，関連研究としてデータストリーム処理エンジンにおける耐障害性保証について概説する．7.1 節ではチェックポインティングによる頑健な耐障害性保証について，7.2 節では近似的な耐障害性保証についてそれぞれ述べる．

7.1　頑健な耐障害性保証

Flink [2] や Samza [3] など多く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 セマンティクスを採用する．これらシステムでは，チェックポインティングによる内部状態のバックアップと入力データの再送・再処理と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一回処理した結果の出力を保証できる．しかし，チェックポインティング処理及び復帰処理においてはファイルストレージや DB との多量のデータ通信が必要となり，処理性能の低下を招く恐れがある．

この問題に対していくつかのシステムでは，データの圧縮による通信量削減や，非同期なバックアップ処理によるシステム全体での処理性能向上といった対策を取っている [11]．しかし，全ての内部状態や入力データをバックアップし，障害復帰時にはチェックポイント時点以降の全入力データの再処理が必要となることには変わらない．センシングデータのような大規模データストリームに対するアプリケーションを念頭に置くと，頑健な耐障害性よりもスループットの向上や迅速な障害復帰が求められると考えられる．

7.2　近似的な耐障害性保証

多くの処理システムでは近似的な耐障害性保証として，at-most-once セマンティクスや at-least-once セマンティクスを採用

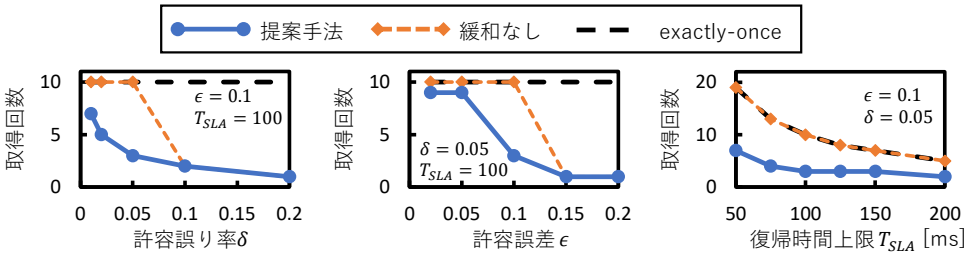

図 6　各パラメータに対するチェックポイント取得回数の変化

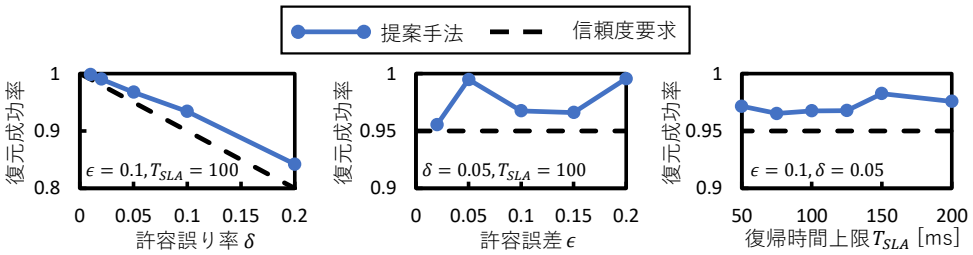

図 7　各パラメータに対する復元成功率の変化

している [12, 13]．これらのセマンティクスでは厳密な耐障害性を保証しないため exactly-once セマンティクスよりもスループットは向上するが，誤差上限などの保証もなく，結果の信頼性に問題がある．

一方，Huang らは結果の信頼性を保証した近似的耐障害性手法を提案した [5]．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，バックアップするデータ量及び頻度を削減し，スループットを向上する．しかしこの手法では，SLA に適した閾値の設定のためには実際のノード構成などを考慮した綿密な事前分析が必要であり，実運用において多くの困難が伴う．

また，Takao らはセンサストリーム処理を対象とした，確率モデルに基づく近似的な耐障害性保証を提案した [14]．この手法は一部のデータのみをバックアップし，障害復帰時には残りのデータを推定し再処理することで，平均・合計・最大・最小問合せに対して直感的な誤差保証を実現する．しかしこの手法は，バックアップデータの削減のみに集中しており，障害復帰処理時間の上限に対する保証がなく，また処理性能の向上に関する言及がないなどの問題点がある．

8　おわりに

本稿では信頼度要求及び復帰時間要求を直感的に指定可能な，近似的耐障害性を保証する手法を提案した．本手法はチェックポイントを効率的に取り入れることで，信頼度要求を満たした近似的復元に必要な再処理対象センサを削減し，復帰時間要求の保証に必要なチェックポイント間隔を緩和できる．また，提案手法の効率性及び妥当性について実験により評価した．

今後の課題として，並列処理における効率的な推定のためのキー割り当ての検討が挙げられる．そのほか，遅延及び損失のあるデータストリームや近似問合せを考慮した近似的耐障害性保証の検討などについても取り組みたい．

謝　辞

本研究は，JSPS 科研費（16H01722, 19K21530）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）の委託業務による．

文　献

[1] “Apache Hadoop.” https://hadoop.apache.org/ (Accessed: December 23, 2019).
[2] “Apache Flink: Stateful Computations over Data Streams.” https://flink.apache.org/ (Accessed: December 23, 2019).
[3] “Samza.” http://samza.apache.org/ (Accessed: December 23, 2019).
[4] “Spark Streaming | Apache Spark.” http://spark.apache.org/streaming/ (Accessed: December 23, 2019).
[5] Q. Huang and P. P. C. Lee, “Toward high-performance distributed stream processing via approximate fault tolerance,” *PVLDB*, vol. 10, no. 3, pp. 73–84, 2016.
[6] D. Reinsel, J. Gantz, and J. Rydning, *Data Age 2025: The Digitization of the World From Edge to Core*. International Data Corporation, 2018.
[7] **第 *1* 部 第 *1* 章 世界と日本の** *ICT*. 平成 30 年版 情報通信白書, 総務省, 2018.
[8] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, “Model-based approximate querying in sensor networks,” *VLDBJ*, vol. 14, no. 4, pp. 417–443, 2005.
[9] “Apache Kafka.” https://kafka.apache.org/ (Accessed: December 23, 2019).
[10] “Mvnorm function | R Documentation.” https://www.rdocumentation.org/packages/mvtnorm/versions/1.0-11/topics/Mvnorm (Accessed: September 24, 2019).
[11] P. Carbone, S. Ewen, G. Fóra, S. Haridi, S. Richter, and K. Tzoumas, “State Management in Apach Flink,” *PVLDB*, vol. 10, no. 12, pp. 1718–1729, 2017.
[12] “Apache Storm.” https://storm.apache.org/ (Accessed: December 23, 2019).
[13] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, “S4: Distributed Stream Computing Platform,” in *2010 IEEE International Conference on Data Mining Workshops*, pp. 170–177, 2010.
[14] D. Takao, K. Sugiura, and Y. Ishikawa, “Approximate Fault Tolerance for Sensor Stream Processing (in press),” in *Australasian Database Conference*, 2020.