D-017

データストリームの集約処理における近似的耐障害性に関する一考察

高尾 大樹　　石川 佳治　　杉浦 健人

名古屋大学大学院情報学研究科

1　はじめに

近年の爆発的なデータ量の増加に伴い，データストリーム処理に注目が集まっている．データストリーム処理は，動的に生成されるデータをリアルタイムに処理し続けるという特徴から，外れ値検出やオンライン機械学習，ネットワークモニタリングなど，幅広いアプリケーションに適用される．

一般的に，内部状態を主記憶上に保持し，処理済みデータを廃棄するデータストリーム処理において，耐障害性の実現は重要な要件の一つである．なぜなら，データストリーム処理は膨大な量のデータをリアルタイムで処理するために並列分散環境下で用いられることが多く，ノードの故障やプロセスダウン，ネットワークの分断といった障害の影響が大きいためである．

耐障害性実現の一つの方針として，処理データや内部状態の定期的なバックアップがある．障害発生時にはバックアップした内部状態や処理データを待機ノード上に再構築することで，誤差のない頑健な障害復帰が可能であり，多くのシステムでこの方法が採用されている [1–4]．しかし，非常に大きなサイズのデータをバックアップするため，バックアップ処理や復帰処理におけるオーバヘッドが問題となる．

そこで，本稿では，OLAP をはじめとする様々なアプリケーションの基本となるデータストリーム集約処理において，統計的な観点からの誤差評価が可能な近似的耐障害性の実現を検討する．

2　バックアップに基づく耐障害性の保証

本章では，バックアップに基づく耐障害性について概説する．特に 2.1 節では基本的なアプローチについて，2.2 節では近似に基づく効率化手法についてそれぞれ述べる．

2.1　基本的なアプローチ

バックアップに基づくアプローチでの基本的なシステム構成を図 1 に示す．システム内の各ワーカノードが持つ内部状態や処理データは，分散ファイルストレージ

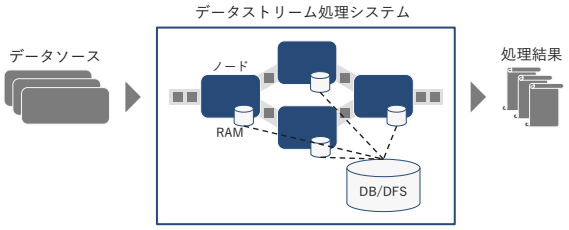

図 1　バックアップに基づいた耐障害性を保証する分散ストリーム処理の基本的なシステムアーキテクチャ

や分散 DB などに定期的にバックアップされる．そして，障害が発生した際にはロールバック処理によって待機ノード上に障害発生前の状態を再構築する．このアプローチでは，バックアップのタイミングや復帰方法などを工夫することで，誤差のない頑健な耐障害性が実現でき，多くのシステムでも頑健な耐障害性をサポートしている．

しかし，頑健な耐障害性を実現するためには，バックアップ処理や復帰処理において分散ファイルストレージや分散 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，既存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が取られている [3]．

2.2　近似に基づく効率化

処理性能低下の問題に対する別のアプローチとして，耐障害性を近似的に保証する Huang らの手法がある [5]．

この手法は，未処理データ数と内部状態の変量に対して閾値を定め，その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の削減とそれにともなうスループットの向上を実現している．バックアップするデータを減らすため障害復帰の際に誤差が生じてしまうものの，閾値によってその誤差の上限を定められる．

この手法では，データの要約処理や近似的問合せ，オンライン機械学習といった，ある程度の誤差を許容できるアプリケーションを想定している．このようなアプリケーションにおいては，近似的な耐障害性のサポートで十分であるため，近似に基づく効率化は有効な方針であるといえる．

近似的な耐障害性をサポートしているシステムは他にも存在する [6,7] が，内部状態や処理データのバックアッ

A Study of Approximate Fault Tolerance in Online Aggregation
Daiki Takao, Yoshiharu Ishikawa, and Kento Sugiura
Graduate School of Informatics, Nagoya University

プは取っておらず，障害発生時は待機ノードへの処理の移譲のみを行う．入力データの再送はできるが，Huang らの手法と違い誤差に対する上限などの保証はなく，結果の信頼性に問題がある．

3 近似的耐障害性の方向性

2.2 節で述べた通り，Huang らの手法では未処理データ数や内部状態の差分に対して閾値を設定することで，誤差保証のある近似的耐障害性を実現する．しかし，この手法にもいくつかの問題点が存在する．

- SLA（Service Level Agreement）に適した閾値の設定が困難である．例えば，未処理データ数に対して適切な閾値を設定するためには，一つのデータの損失が最終結果に与える影響の綿密な分析が必要である．しかし，耐障害性が重要になる複雑なアプリケーションほど，そのような分析は困難となる．
- 適用可能なアプリケーションが限定的である．未処理データ数が閾値を超えた際にバックアップを取るだけでは，例えば，データの分布に偏りがある場合やデータの重要度が異なる場合などにおいて適切な誤差保証が難しくなる．
- 対象とする障害の種類が限定的である．この手法でサポートしているのはノード障害やプロセスダウンといった障害に対する耐障害性であり，データ遅延などについては考慮されていない．

以上から，近似的耐障害性に関する研究の方向性として，ノード障害時の誤差保証の方法や適用可能範囲の拡張と，データ配送に関する耐障害性の実現の大きく二通りが考えられる．以降ではそれぞれの方向性について考察する．

3.1 ノード障害に対する近似的耐障害性

ノード障害に対する耐障害性に関する問題点については，近似的問合せ手法の応用が考えられる．つまり，一般的な近似的問合せ手法と同様に，サンプリングによってバックアップ対象のデータを選択し，信頼区間などの形で誤差保証を行うという方針である．

信頼区間のような形で誤差を保証できれば，ユーザは複雑な分析をせずに，SLA に適したシステム設定が可能となる．また，データの偏りなどを考慮したサンプリング手法を用いた近似的問合せに関する研究も報告されており [8]，このような手法を応用することでより広い種類のアプリケーションへの適用も期待できる．

3.2 データ配送に対する近似的耐障害性

入力データに付与された識別 ID や時刻印に基づいてデータの処理状況を把握することで，データ配送に対する耐障害性を保証するシステムが存在する [2, 9]．しかし，これらのシステムでは頑健な耐障害性を想定しており，データの遅延に対してはデータ到着時の処理結果修正，データの損失に対してはバックアップからの再送といった方針を取っている．

データ配送に対する近似的な耐障害性の実現方法としては，データの遅延や欠損による誤差の評価や，過去のデータの傾向に基づいた補間などが考えられる．

4 まとめと今後の課題

本稿では，データストリーム処理に対する耐障害性実現における問題点と近似的なアプローチによる解決について議論した．今後の課題としては，今回議論した解決策について具体的な手法の考案とその実現が挙げられる．

謝辞

本研究の一部は科研費（16H01722）による．

参考文献

[1] S. A. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, and R. H. Campbell, “Samza: Stateful Scalable Stream Processing at LinkedIn,” in *Proc. VLDB Endowment*, vol. 10, pp. 1634–1645, 2017.

[2] T. Akidau, A. Balikov, K. Bekiroğlu, S. Chernyak, J. Haberman, R. Lax, S. McVeety, D. Mills, P. Nordstrom, and S. Whittle, “MillWheel: Fault-Tolerant Stream Processing at Internet Scale,” in *Proc. VLDB Endowment*, vol. 6, pp. 1033–1044, 2013.

[3] P. Carbone, S. Ewen, G. Fóra, S. Haridi, S. Richter, and K. Tzoumas, “State Management in Apach Flink,” in *Proc. VLDB Endowment*, vol. 10, pp. 1718–1729, 2017.

[4] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, “Discretized Streams: Fault-Tolerant Streaming Computation at Scale,” in *Proc. SOSP*, pp. 423–438, 2013.

[5] Q. Huang and P. P. C. Lee, “Toward High-Performance Distributed Stream Processing via Approximate Fault Tolerance,” in *Proc. VLDB Endowment*, vol. 10, pp. 73–84, 2016.

[6] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, “S4: Distributed Stream Computing Platform,” in *KDCloud*, pp. 170–177, 2010.

[7] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy, “Storm @Twitter,” in *Proc. SIGMOD*, pp. 147–156, 2014.

[8] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding, “Quickr: Lazily Approximating Complex AdHoc Queries in BigData Clusters,” in *Proc. SIGMOD*, pp. 631–646, 2016.

[9] R. Ananthanarayanan, V. Basker, S. Das, A. Gupta, H. Jiang, T. Qiu, A. Reznichenko, D. Ryabkov, M. Singh, and S. Venkataraman, “Photon: Fault-tolerant and Scalable Joining of Continuous Data Streams,” in *Proc. SIGMOD*, pp. 577–588, 2013.