D-005

センサデータのための分散サンプリングストレージにおける書き込み処理

Writing Process of Distributed Sampling Storage for Sensor Data

佐藤 浩史 †‡ Hiroshi SATO　東島 由佳 † Yuka HIGASHIJIMA　中村 元紀 † Motonori NAKAMURA　松村 一 † Hajime MATSUMURA　小柳 恵一 ‡ Keiichi KOYANAGI

1　はじめに

今後 10 年で Internet of Things が発達し，実空間に遍在する各種のセンサがネットワークに接続し，人や物，そして環境の膨大な情報が絶え間なく流通すると言われている [1][2]．このような膨大かつ連続的なデータは，瞬間値である個々のレコードを独立に見るだけでなく，データの変化や全体の特徴量などの統計的な傾向を捉えることでより価値が増すと考えられる．また，それらの算出には確率解や近似解法を使うことが多い [3] ため，サンプリング処理を施すことが一般的であり，レコード個々の永続性はさほど重要ではないと考えられる．そこで我々は，センサデータの特性を踏まえ，永続性を緩和したシンプルな分散ストレージ技術の研究を進めている [4][5]．このストレージはデータの統計的利用を前提としており，サンプリングの特性を利用して規模拡張性ならびに可用性・障害耐性を高めたアーキテクチャが特徴である．本稿ではこのストレージの概略を述べ，一時的なサーバダウンによる書き込み失敗時のリカバリ処理について提案する．

2　分散サンプリングストレージ

2.1　要件

増え続けるデータおよびアクセスへ対応するために，規模拡張性および可用性は必須である．一方，既に述べたように，個々のレコードごとの永続性は要求されない．但し，指定された特徴量を正しく算出するのに十分なデータを，偏りなく保持していなければならない．また，特徴量の種類および要求される精度は利用するアプリケーションにより様々であるため，一般のデータストリーム処理で見られるようにデータの要約のみを保持する方法は好ましくない．データそのものを保持する必要がある．しかしその一方，常に全データを返すことは，ストレージおよびネットワークにかかる負荷を考慮すると，得策ではない．要求された精度とその処理によりかかる負荷が，トレードオフの関係になることが望ましい．

2.2　アプローチ

複数のデータベース（データサーバ）にデータを水平分割する分散構成を取る．その際，各データサーバ間には固定された上下関係を持たせず，互いの状態も監視せず，それぞれ独立にクライアントからのアクセスを受け付ける．クライアントはランダムに選択されたデータサーバへレコードを書き込む．これにより，各データサーバには全レコードに対するランダムサンプルが独立に蓄えられる．つまり，ストレージ自身にランダムサンプリング機能が内包されることになる．ここでラウンドロビンではなくランダムに選択する理由は，各サンプルにおいてデータの周波数成分が消失することを防ぐためである．このようなストレージを**分散サンプリングストレージ**と呼ぶことにする．

2.3　基本アーキテクチャ

本ストレージは管理サーバと複数のデータサーバから成り，クライアントはその両者にアクセスしてレコードを書き込み，また，データの統計量を得る (図 1)．

管理サーバはデータサーバの状態管理を主な役割とし，その情報をレコード読み書き時のデータサーバ選択に利用する．自身の負荷を抑えるためレコードの読み書きは行わず，データサーバやクライアントが直接読み書きを行う．各データサーバは独立に動作し，クライアントからのアクセスにそれぞれ対応してレコードを蓄積・検索する．

レコードを書き込む際のデータサーバの選択について，管理サーバが行う方法とクライアントが行う方法の二通りがある．管理サーバが行う場合，クライアントは書き込みの都度管理サーバにアクセスし，管理サーバが，正常に稼動しているデータサーバから 1 台をランダムに選び[*1]，そのアドレスをクライアントに伝え，クライアントがデータサーバへレコードを書き込む．クライアントが行う場合，管理サーバは稼動しているデータサーバのリストを公開し，クライアントはそのリストからデータサーバをランダムに選び，レコードへ書き込む．どちらの場合も，管理サーバへのクライアントからのアクセス頻度が高くなると管理サーバの負荷が大きくなってしまう点が問題であるが，データサーバの選択をクライアントが行う場合は，データサーバのリストをクライアントが一定期間キャッシュすることで，管理サーバへのアクセス頻度を下げることができる．本稿ではこちらの方法を採用する．この方法により生じる課題と解決方法については 3 節で詳しく述べる．

統計量を算出する際は，クライアントはクエリとして，統計量の種類と要求する精度，そして検索条件を管理サーバに伝える．管理サーバは要求精度から必要なサンプルサイズを見積り，負荷分散を考慮した上でデータサーバを必要な数だけ選択し，うち 1 台をレコードを集約して統計量を計算する代表サーバに指定し，代表サーバにクエリと他のデータサーバのアドレスリストを渡す．代表サーバは検索条件を各データサーバに投げ，適合するレコードを集め，統計量の計算を行い，結果をクライアントに返す[*2]．サンプルサイズの見積りとデータサーバ数の決定については [5] を参照されたい．

2.4　効果

本アーキテクチャに基づくシステムでは，以上の方式によりサンプルサイズの制御を行い，精度とシステム負荷とのトレードオフも成立させる．各データサーバは他のサーバの状態を持

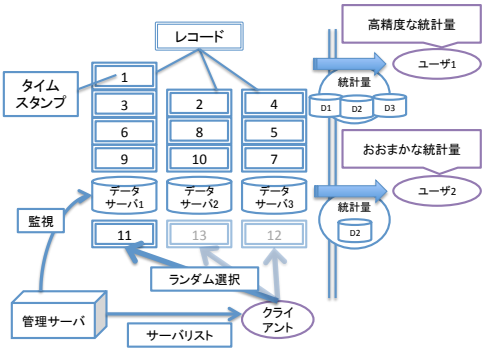

図 1　アーキテクチャ

† 日本電信電話 (株), NTT　‡ 早稲田大学, Waseda Univ.

[*1] 一度に n 個のレコードを書き込む場合は，n 台のデータサーバを重複を許してランダムに選ぶ．

[*2] 統計量ではなくサンプルそのものを返すことも可能である．

つ必要がないため，高い規模拡張性が期待できる．書き込み時の無作為性により，各データサーバには等質なデータがそれぞれ蓄積されることから，一部にサーバダウンが発生してもデータの消失に特定の偏りは発生せず，他のサーバが自然に補填できる．従って，その影響は最高精度の若干の低下のみで済むので，データのレプリケーションを行わずともフェイルソフトなシステムが実現できる．

3 書き込み処理

3.1 データサーバリスト

管理サーバは定期的にデータサーバの状態を監視し，その結果を内部リストとして保持する．その内部リストから，現在クライアントのアクセスに対応可能なサーバの一覧を外部リストとして有効期限付きで公開し，定期的に更新する．有効期限は次の更新時刻である．クライアントは初回の書き込みアクセス時に管理サーバから外部リストをキャッシュし，有効期限までは管理サーバへはアクセスせずに，そのキャッシュを参照しデータサーバの選択を行う．そして，有効期限を過ぎた後の初回のアクセス時にキャッシュを破棄し，改めて管理サーバからリストを取得しキャッシュする．

このように外部リストをクライアントに一定期間キャッシュさせるメリットは，リスト取得たのめのアクセスが減ることによる管理サーバの負荷の低減である．デメリットは，データサーバの実際の状態と外部リスト上の状態に不整合が生じ易くなり，その結果，クライアントが選択したデータサーバが実際には稼動しておらず，書き込みに失敗する可能性が高くなることである．

3.2 書き込み失敗時のリカバリ

クライアントの選択したデータサーバのダウンにより書き込みを受け付けなかった場合，対処方法としてすぐ思い付くことは，該当サーバを除いたリスト上のサーバ群から改めて選択し直すことである．しかし，分散サンプリングストレージではそれは好ましくない．クライアントの判断による再選択を許可すると，サーバダウンを考慮するクライアントとしないクライアントが同時に存在することになり，各データサーバに蓄積されるレコード群に管理サーバの想定外の偏りが生じ，サンプルの精度を保証できなくなるためである．対処方法は大きく二つある．一つは，書き込みに失敗したレコードはそのまま破棄するという方法である．この場合，データサーバへの振り分けの観点での偏りは防ぐことができるが，ほんの一時的なサーバダウンでもその期間の該当サーバのレコードを全て捨てることになり，好ましくない．もう一つは，データサーバの当初の選択を生かした上でレコードを一時的に別途保管する方法である．我々は以下のように後者の方法を採用する．

通常のデータサーバに加えて，代替サーバを設ける．データサーバのダウンによりレコード書き込みに失敗したクライアントは，管理サーバにその旨を伝え，管理サーバから指定された代替サーバにアクセスし，当初選択していたデータサーバの識別子とともにレコードを書き込む．代替サーバはデータサーバの識別子毎にテーブルを準備し，そこにレコードを挿入する．管理サーバはデータサーバの監視を続け，データサーバが復旧した時点でその旨を代替サーバに伝える．代替サーバはそれを受けて，該当するテーブルの全レコードをデータサーバへ書き戻し，自身からテーブルごと削除する．その処理が終了した後に，管理サーバは内部リストを更新し，定時に公開する (図 2)．

以上により，外部リストの公開後，次に更新されるまでのあいだにデータサーバがダウンし，そのデータサーバがクライアントにより選択された場合，レコードは代替サーバに書き込まれる．データサーバの復旧より前に外部リストの更新があれば，それ以降はクライアントが正しくデータサーバの稼動状況を把握するので，ダウンしているデータサーバにレコードが書き込まれることはない．よって，代替サーバへレコードが書き込まれ続ける期間は高々外部リストの更新間隔なので，外部リストの更新頻度やデータサーバの監視頻度を適切に設定すれば，代替サーバの負荷はそれほど高くならない．

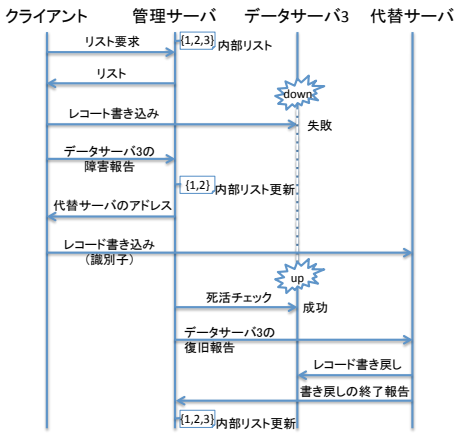

図 2 リカバリのシーケンス図

3.3 クライアントによる死活監視

以上のリカバリ処理により生まれる副次的な効果として，管理サーバの死活監視の負荷の低減がある．クライアントは書き込みに失敗した際に管理サーバへ報告を行うので，管理サーバは自分の気付いていないデータサーバのダウンを知ることができる．この仕組みを利用してクライアントに死活監視を分担させ，その分自らが行う定期監視の頻度を下げることで，リカバリ処理に伴う管理サーバへのアクセスによる負荷の増大をキャンセルできると考えている．

4 おわりに

センサデータの統計的利用を前提とした分散サンプリングストレージのアーキテクチャについて説明し，データサーバダウン時のリカバリ処理方式を提案した．この方式により，管理サーバの負荷を低く抑えながら，データサーバの一時的なダウンがあっても意図しないデータの偏りを生じさせずにストレージを運用し続けられることを示した．現在我々はプロトタイプの実装を進めており，今後，実験により機能評価および性能測定を行う予定である．

参考文献

[1] “Internet of Things in 2020: roadmap for the future,” European Technology Platform on Smart Systems Integration final report, May 2008.

[2] M. Balazinska, A. Deshpande et al., “Data management in the worldwide sensor web,” IEEE Pervasive Computing, vol.6, no.2, pp.30–40, Apr. 2007.

[3] 有村博紀, “大規模データストリームのためのマイニング技術の動向, ” 信学論（D）, vol.J88-D-I, no.3, pp.563–575, Mar. 2005.

[4] 佐藤浩史, 井上武 他, “センサデータマイニングのための分散サンプリングストレージの提案〜統計量の基本的な挙動解析を中心に,” 信学技報, vol. 109, no. 449, IN2009-183, pp. 235–240, 2010 年 3 月.

[5] 佐藤浩史, 井上武 他, “センサデータのための分散サンプリングストレージとそのサンプルサイズ制御手法,” 第 9 回情報科学技術フォーラム (FIT2010), D-022, 2010 年 9 月.