一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
CPSY2018-108,DC2018-90（2019-03）

分散データ分析における省リソース・高効率なソース側重複除去手法

五木田 駿† 一角 健人† 久保田 真† 福山 訓行†

†株式会社富士通研究所 〒211-8588 神奈川県川崎市中原区上小田中４－１－１
E-mail: †{gokita.shun, ikkaku.kento, kubota.makoto, noriyuki}@fujitsu.com

あらまし 近年，IoT デバイスにより生成される大量データを分析・活用することで，企業の業務効率化やサービス品質向上などが期待されている．IoT デバイスは一般的に計算リソースが乏しく，データをクラウドに集約して分析することが多いが，一方で地理的に分散されているためにネットワーク帯域も細く，すべてのデータを集約することが難しい．本研究は，エッジ側の計算リソースを抑えながら重複除去処理を行い，かつチャンク単位を適応的に制御して重複検出効率を向上させることでネットワーク転送量を効果的に削減する手法を提案する．
キーワード 分散データ分析，重複除去，モビリティ IoT

An Update-Range-Aware Source Deduplication Method with Low-Resource and High-Efficiency for Geo-Distributed Data Analysis

Shun GOKITA† Kento IKKAKU† Makoto KUBOTA† and Noriyuki FUKUYAMA†

†Fujitsu Laboratories LTD. 4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa, 211-8588 Japan
E-mail: †{gokita.shun, ikkaku.kento, kubota.makoto, noriyuki}@fujitsu.com

Abstract In recent years, business efficiency and service quality improvement are expected by analyzing and utilizing the large amount of data generated by the IoT device. In general, IoT devices are poor in computing resources to analyze data, but on the other hand, because of the geographical distribution, the network bandwidth is also narrow and it is difficult to aggregate all the data in the cloud. In this research, we propose a method that effectively reduces the amount of network transfer by performing deduplication processing while saving computing resources on edge, and adaptively controlling chunk units to improve duplication detection efficiency.
Keywords geo-distributed data analysis, deduplication, mobility IoT

1. はじめに

近年，IoT デバイスの普及が急速に進んでいる．最近の研究によれば，IoT デバイスの数は 2016 年の 58 億から，2021 年には 137 億にまで増えると予想されている[1]．これらのデバイスはセンサーデータ，画像・映像・音声等のメディアファイル，動作ログなどの膨大なデータを日々生み出し続けており，そのデータ量は 2016 年の 218 ゼタバイトから，2021 年には 847 ゼタバイトになるとも言われている．その膨大な IoT ビッグデータを分析し，企業の業務効率化やサービス品質向上などに活かそうとする動きが活発になっている．例えば，世界最大級の輸送業者 UPS は，同社の 8 万台以上の配送車に取り付けたセンサーで速度・燃費・走行距離・停止回数・エンジンの状態など 200 種類以上のデータを取得し，それらを分析・活用することでアイドリング時間や燃料消費を削減し，経費削減・効率改善・環境負荷軽減に役立てている[2]．

このような IoT ビッグデータの利活用事例の多くはクラウドをコンピューティング基盤として行われており，IoT デバイスにより生み出される大量のデータを一旦クラウドに集めてから分析処理を行う．しかし，前述の通り IoT データが肥大化するに従い，デバイスからクラウドへ全てのデータを集約することは現実的に不可能であることが認識されるようになった．そこで，全てのデータをクラウドに送って処理するのではなく，IoT デバイス側でできる処理は，IoT デバイス側で行い，最終的に必要なデータのみをクラウドに送ることでクラウド側でしかできない処理をより高速に行うというエッジコンピューティングの概念が提唱されるようになった．エッジコンピューティングの例として，IoT デバイスで記録された動画などの非構造化データからエッジで特定の情報のみを抽出して構造化し，クラウド側で集計するような処理を行う場合が考えられる．この場合，非構造化データを全て集約する場合に比べてデータ転送量を大幅に小さくすることが可能である．しかし，前述のように IoT デバイスの数は爆発的に増加しており，このような一次処理による削減だけではなくさらなる転送量削減を考える必要がある．

エッジ-クラウド間のデータ転送量を削減する手法として，圧縮や差分転送，重複除去転送技術がある．圧縮技術に関しては様々な手法が研究されており，例えば辞書式圧縮方式では頻出パターンを辞書に登録し，そのパターンが出現するたびに辞書のインデックスに置き換えることでデータ量を削減する仕組みであるが，

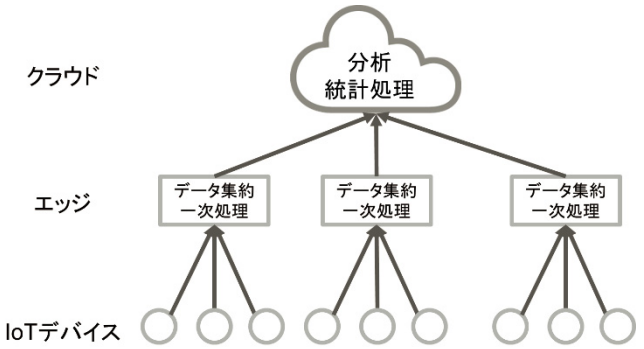

図 1：本稿で前提とする分散データ分析システム構成

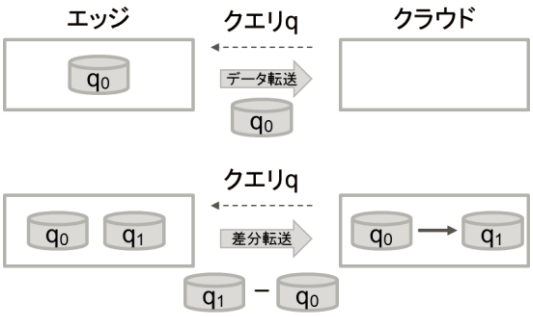

図 2：差分転送方式の例

対象データによってはほとんど効果がない場合がある．差分転送技術は，一度送信されたデータがその後更新され，再度転送する場合に更新部分のみを転送する技術である．この手法では，前回送信したデータとの差分を取るために送信元・受信先双方が送信されたデータをキャッシュしておく必要がある．エッジは一般的に計算リソースが乏しく，送信済みデータ全体をキャッシュしておくのは現実的に難しい．重複除去転送技術は，エッジ側でのキャッシュは不要だが，重複除去効率を上げるためにはエッジ側にも十分な計算リソースが必要である．

本研究では，この重複除去転送方式の発展として，エッジ側の計算リソースを抑えながら効率的に差分を検出し，重複部分を除去して転送する技術を開発した．CAN データを用いたシミュレーションでは，データ転送量を最大で 64%削減することができた．

以下に本稿の構成を示す．2 章では本提案手法が想定する分散データ分析システムについて説明する．3 章で本提案手法について述べ，4 章ではその評価結果を示す．5 章では分散データ分析におけるデータ転送量の削減に着目した関連研究を紹介する．6 章ではまとめと今後の課題について述べる．

2. 想定するシステムとその課題

本稿における提案手法は，ネットワークを経由して大容量データを転送するシステム全般に通じる課題を解決するものであるが，本章では前提としているシステム構成を説明する．また，これらのシステムで一般的に用いられるデータ転送量削減手法について示し，その課題について述べる．

2.1. システム全体構成

本研究では，広域に分散された IoT デバイスが生成するデータを，エッジと呼ばれる端末で集約，および一次処理し，クラウドにデータを集約・分析するシステムを想定している（図 1）．エッジは CPU，メモリ，ストレージ，ネットワーク I/F を持つ一般的なコンピュータ構成を想定しているが，例えばモビリティ分野への適用を想定した場合，カーナビ等の車載器となるためスペックは高くないことを前提としている．クラウド側では，計算リソースは要求に応じてスケールできるものとして，特に制約は考えない．

代表的な使用例は以下である．まずデータ分析者が作成した分析ロジックに基づきエッジへのデータ要求を行う．エッジ側ではその端末にあるデータだけでできる処理を一次処理として行ってその結果をクラウドへ集約する．クラウドでは，他のエッジから集約したデータと併せてさらなる分析処理を行い，最終的な分析結果を得る．このような一連の流れを想定している．

想定するデータ分析の種類としては，生成されるデータを逐次的に分析して監視するようなストリーミング処理ではなく，必要になった時点でまとめて処理をするバッチ処理や，対話的にデータ分析を行うアドホック処理を想定している．そのため，IoT デバイスから生成されるデータはエッジで一定期間保存する前提であり，そのための十分な記憶領域が必要である．

2.2. データ転送量の削減方式

エッジコンピューティングを前提とした分散データ分析では，エッジで一次処理を行った全てのデータをクラウドに集約して分析することは難しい場合もあるため，さらなる転送量削減を考える必要がある．ここでは代表的なデータ転送量削減方式として，差分転送方式と重複除去転送方式について示す．

2.2.1. 差分転送方式

差分転送技術は，一度送信元から受信先に送信されたデータについて，送信元で更新されてから再度送信する場合に，更新部分のみを転送し受信先で更新後のデータ全体を復元する技術で，http のプロトコル等に採用されている[3]．この技術を分散データ分析に適用した場合の例を図 2 に示す．あるデータについて，クラウドからエッジに対して分析要求 q を出した場合を想定する．初回要求時では，エッジでは要求された演算を行い，その結果 q_0 をクラウドへ返すが，エッジ・クラウドともこの結果を保存しておく．次回同じ分析要求が来た場合，エッジでは再び要求された演算を行って結果 q_1 を得るが，クラウドへ返すのは以前保存した結果との差分 q_1-q_0 である．その差分と，クラウドでキャッシュしてある q_0 から，更新後のデータ q_1 を復元する．以上のようなプロトコルにより，更新サイズが小さい場合にはデータ転送量を大幅に小さくすることが可能である．

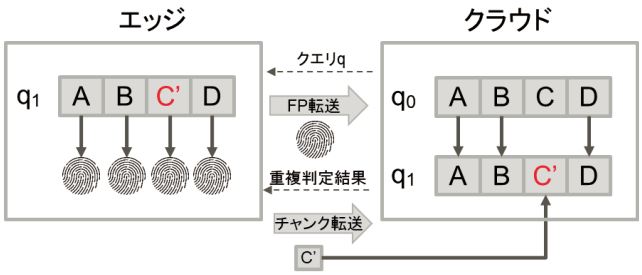

図 3：重複除去転送方式の例

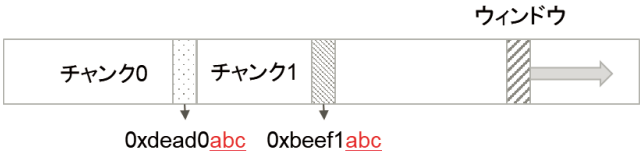

図 4：CDC による可変長チャンキングの例．
この例では下位 12bit が 0xabc であればチャンク区切りと判定している．

2.2.2. 重複除去転送方式

重複除去技術を用いたデータ転送量削減手法は，転送しようとしているデータがすでに転送した事があるデータかどうかを送信元,受信先のどちらかが判定し，重複だと判定された場合はその部分を除去して差分のみを転送する技術である．重複除去技術はストレージシステムをはじめ ICT インフラで広く使われており，商用製品でも各社様々な手法を採用しているが，本節ではリモートファイルの同期コマンドである rsync[4]に採用されているプロトコルを紹介する（図 3）．

まず送信対象データをチャンク単位に区切り，チャンク毎にハッシュ値(以後フィンガープリントと呼ぶ)を取る．データを送信する前にこのフィンガープリントをクラウドに転送し，クラウド側で保持しているチャンクのフィンガープリントと比較することで重複しているかどうかを判定する．フィンガープリントが一致する場合はそのチャンクが重複していると判断し，転送が不要となる．フィンガープリントが異なる場合は重複していないチャンクであると判断し，そのチャンクをクラウドへ転送する.以上のプロトコルにより，データ転送量を削減する仕組みである．

2.3. 従来手法のエッジ適用における課題

差分転送技術は，以前に転送したデータをエッジ，クラウド双方がキャッシュしておく必要がある．例えばエッジに蓄えられた 100GB のデータに対して，異なる 100 通りの分析要求があり，各要求につき 1GB のデータをクラウドへ転送する場合，転送データのキャッシュを 100GB 持たなければならならず，エッジに蓄えられるデータ量を圧迫する．そのため，転送量削減のためだけに転送済みのデータ全体をエッジでキャッシュしておくのは現実的に難しい．

一方，重複除去を用いたデータ転送量削減では，チャンク単位の重複判定・転送になるためにデータの転送効率は差分転送技術よりも悪くなるが，送信元で前回結果のキャッシュが不要なため，エッジ-クラウドを想定した場合に適している．

重複除去転送方式では，データの種類やチャンクの区切り方が性能に大きな影響を与えることが知られている[5]．チャンクの区切り方については大きく分けると固定長方式，可変長方式の 2 通りがある．

固定長方式は計算負荷が軽いが，間に新たなデータを挿入するなどした場合，他のデータの更新がなくともチャンクの区切り単位が変わって重複と判定されなくなるチャンクが増え，重複除去効率が悪くなる．

可変長方式は，主にデータ挿入への対策技術を中心に発展してきた方式である．例えば LBFS（Low Bandwidth Network File System）[6]等で採用され広く使われている CDC（Content Defined Chunking）という可変長チャンキング手法は，データを先頭から順にスキャンし，Anchor と呼ばれる特定のパターンが現れる度にチャンクの境界とする手法である．具体的には，固定長のウィンドウ単位でハッシュ値を計算し，その値の下位数 bit が Anchor と一致していればチャンク境界とするという処理を，1 バイトずつウィンドウをずらしながら繰り返し行う（図 4）．CDC は間にデータが挿入される場合でもそれに追従してチャンク分割が可能になるため，データの挿入に弱いという固定長方式の課題を解決しているが，計算量が多いという別の課題がある．

固定長方式，可変長方式に共通する別の課題として，チャンクサイズに対して更新サイズが小さい場合，転送されるチャンク内にも重複データが多く含まれるため重複除去効率が悪化するという問題がある．一方，重複除去効率を上げるためにチャンクサイズを小さく設定すると，チャンク毎に生成されるフィンガープリントやその他メタデータのオーバーヘッドの合計サイズが大きくなってしまう．このようなトレードオフ関係のため，特にネットワーク帯域が制限されるエッジ-クラウド環境では，チャンク内の重複もできる限り減らしたい．さらに CDC のようにデータ挿入耐性もあり，かつ固定長方式のように計算負荷が軽いチャンキング手法が求められる．

3. 提案手法

前章で述べたように，差分転送方式，重複除去転送方式ともに，IoT ビッグデータを想定した分散データ分析システムへの適用には課題が残る．本章では，本稿で提案する省リソースで高効率なソース側重複除去手法について示す．

3.1. 全体構成

図 5 に本手法の全体構成を示す．まず分析ロジックに基づき，クラウドからエッジに対してデータ要求を行う．エッジ側ではデータソースから該当データをク

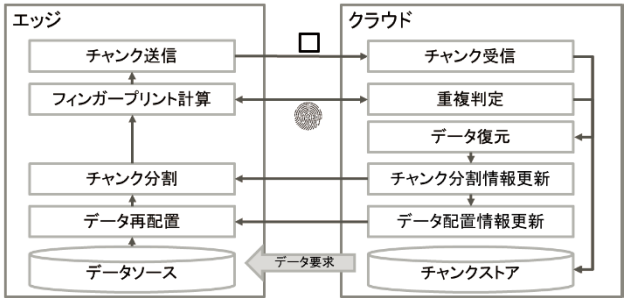

図 5：提案手法の全体構成

ラウドに送る必要があるが，重複除去転送の要領でチャンクに分割してフィンガープリントを計算し，それをクラウドに送信して重複判定を行う．フィンガープリントの計算に用いるハッシュ関数は，実用上はほぼ衝突することがないように暗号学的なハッシュ関数，例えば SHA-1 や SHA-256 ハッシュを用いる．ここでのチャンク分割はクラウド側で動的に決定し，その情報に基づきチャンク分割を行う．詳細は次節以降にて示す．重複と判定されなかったチャンクはクラウドへ転送し，重複チャンクと合わせて元データが復元される．この復元されたデータを元に次回のチャンク分割の情報を更新し，エッジ側に返す．以上の流れを，クラウドからエッジに対するデータ要求の度に行うことで，エッジ側での計算リソースを節約しつつ，データ転送量を削減している．

3.2. 差分による可変長チャンキング

前章で述べたように，重複除去効率を上げるためには更新頻度が低い領域は大きなサイズのチャンクとして設定し，更新頻度が高い領域はなるべく更新範囲と一致するようなチャンクとして区切られていることが望ましい．さらにエッジでは計算負荷が高い処理は避けたい．本手法ではこのチャンク区切りを決める方針として，IoT ビッグデータ分析で更新されるデータは空間的局所性が高い，すなわち「前回更新された範囲は今後も更新される可能性が高く，更新されなかった範囲は今後も更新されない可能性が高い」との仮説を立て,それに従ったチャンク区切りを行うこととした．この仮説の有効例として，車載デバイスから集めた CAN データをエッジで構造化してカラム形式でクラウドに送る例を示す．この例では，エンジンの回転数や位置情報などは頻繁に変わっていくが，外気温などは短時間のうちには変わらないため，更新箇所には局所性があると言える．

上記方針に基づき，本手法ではチャンクの区切りを前回のデータ要求結果の差分を用いて行い，前回から更新された範囲を新たなチャンクとして設定する．その例を図 6 に示す．この例ではチャンク C，G はチャンク内の更新部分を新たなチャンク単位とし，更新されなかった部分は隣接チャンクと結合する．このように決定した新たなチャンク区切り情報をエッジ側に送

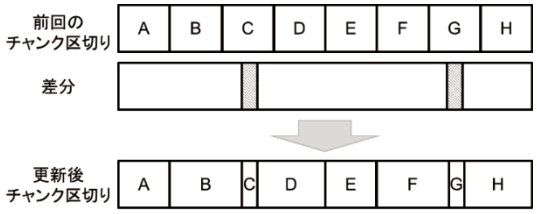

図 6：差分によるチャンキングの例．この例ではチャンク C、G の一部のみ更新があるため，その更新部分を新たなチャンク単位とする．

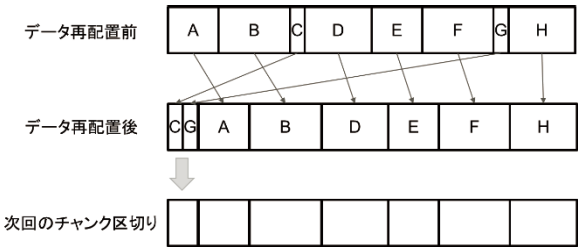

図 7：データ再配置によるチャンク再構成の例．この例ではチャンク C と G が閾値以下のサイズの場合，これらのチャンクを先頭に集約して 1 つの新たなチャンクとして再構成する．

信し，次回のデータ要求ではそのチャンク区切り情報を元にフィンガープリントを生成する．

3.3. データ再配置によるチャンク再構成

前節で示した差分による可変長チャンキングの手法では，更新範囲がごく小さい場合は小さなチャンクができてしまう可能性がある．チャンクサイズがフィンガープリントサイズに近づくと，小さなチャンクが大量にある場合にはデータ転送量に占めるフィンガープリント転送量の割合が高くなり，重複判定の効果が薄れてしまう．そのため，一定サイズ以下のチャンクを物理的に一箇所にまとめて新たなチャンクとするデータ再配置および，チャンク区切り変更を行う．その例を図 7 に示す．この例ではチャンク C と G が閾値（例えばフィンガープリントサイズの 2 倍）以下の場合，これらのチャンクを集約して 1 つの新たなチャンクとして再構成する．

データ再配置情報を含めた，チャンク区切り情報をエッジ側に送信し，エッジでは次回のデータ要求にてこのチャンク区切り情報とデータ再配置情報を用いてチャンク分割を行い,フィンガープリントを生成する．

上記の処理フローにより，エッジ側ではデータのチャンク分割とフィンガープリントの計算という処理のみで済み，さらにこれらの処理で使った中間データを全て破棄しても問題ないため，使用する計算リソースは固定長方式と同様に少ない．また，チャンク区切りも更新範囲に応じてクラウド側で動的に更新していくため，局所性が高い場合はチャンク内での重複を削減できる．データ挿入に関しては，毎回データ長が変わ

表 1：従来方式と提案方式の比較

方式	利点	欠点
差分転送	・転送効率が最高	・膨大なキャッシュ容量
固定長 重複除去	・転送元の計算が軽い	・データ挿入に弱い ・チャンク内重複
可変長 重複除去	・データ挿入に強い	・計算負荷が重い ・チャンク内重複
提案手法	・転送元の計算が軽い ・データ挿入耐性あり ・チャンク内重複削減	・更新の局所性に依存 ・連続データ挿入に弱い

る場合には弱いが，一度データ挿入され次回からは同じサイズの場合には 2 回目からは初回の更新に追従できる．以上を踏まえ，従来手法と提案手法の比較を表 1 に示す．

4. 評価

本章では，本稿で提案する手法の評価方法とその結果について示す．

4.1. 評価方法

本提案手法の適用先の一例としてモビリティ分野を想定し，モビリティ IoT データを用いた評価を行った．想定するシナリオとしては，車載の計測器から取得した CAN（Controller Area Network）データを異常検知等のために一定間隔（例えば 1 時間間隔）でクラウドへ転送する場合を考える．評価に用いたのは，車両に搭載した計測器から取得した実データである．走行場所等が異なる 14 種類のデータ（dataset A～N）を用いた．各データは同一のフォーマットに構造化されており，車両 ID，タイムスタンプ，速度などの情報が高頻度に記録されている．1 時間で蓄積されるデータサイズは車両 1 台あたり約 13MB になる．

本提案手法は，前回の転送から更新された割合の影響を大きく受けるため，本データセットの更新率についても示す．更新率が最も高いものが 23%（dataset F），最も低いものが 3%（dataset G）であり，概ね 10%前後であった．またこれらの更新箇所の時系列変化の一例を図 8 に示す．

これらのデータセットを用い，シミュレーションにより転送量の削減効果を確認した．シミュレーションに用いたパラメータを表 2 に示す．比較対象としたのは，エッジ側で同程度の計算リソースを使用する固定長の重複除去方式である．フィンガープリントの生成に使われるハッシュ関数は SHA-1 を想定して 160bit とし，従来手法のチャンクサイズは 1KB 固定，提案手法の場合は 64B を閾値としてそれ以下のチャンクを集約し，最大 1KB までを 1 チャンクとしている．

4.2. データ転送量の削減効果

図 9 に各データセットでの転送量を，データ更新量を 1 とした相対値で示す．最も転送量削減効果の高かった dataset G では従来手法と比較して 64%削減の効果が見られた．逆に最も効果が低かった dataset B でも 47%の削減効果があり，平均で 55%の削減効果が得られた．最も転送量削減効果の高かった dataset G は前節で述べた通り最も更新率が低く，チャンクサイズに対しての更新分も小さかったため，固定長チャンクの重複除去転送方式では無駄な転送が多く本手法の効果が大きかったと考えられる．一方，最も削減効果の低かった dataset B は更新率こそ 14%で今回用いたデータセット全体の中では中位だが，更新箇所の局所性が他のデータセットと比べて弱く，提案手法の効果が低かったと考えられる．

また，データ転送量の内訳として，データ本体とフィンガープリント，その他のメタデータの割合の比較を図 10 に示す．提案手法はデータ再配置の効果を確認するため，データ再配置有無の二通りで比較している．従来手法と比較して，データ本体は平均 63%と大きく削減しているが，フィンガープリントサイズとその他のメタデータサイズはそれぞれ 116%増加している．これは従来手法に比べてチャンクの分割数が増えているためと考えられる．また，データ再配置の有無で比較した場合，データ再配置により平均で 8%，最大で 14%（dataset C）の削減効果が見られた．データ再配置有りの場合は，集約されたチャンクのうち更新されていないチャンクが含まれると不要なデータの転送が増えるため，データ本体の転送量が平均で 4%の増加が見られるが，フィンガープリントサイズおよびその他のメタデータサイズを平均 36%削減しており，ト

表 2：評価に用いたパラメータ

パラメータ	従来手法	提案手法
フィンガープリントサイズ	160bit	160bit
チャンクサイズ	1KB	64B～1KB
メタデータサイズ	8B	8B

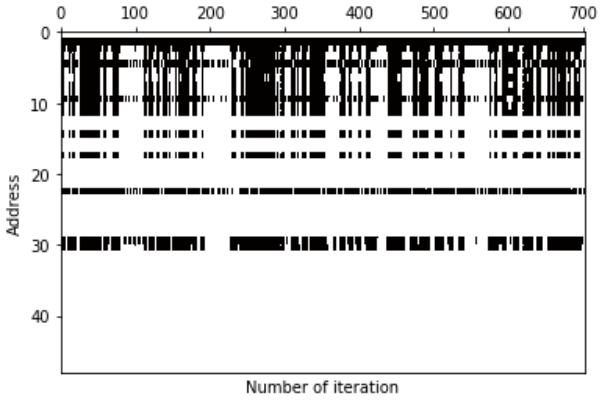

図 8：更新範囲の時系列変化の例(dataset C)．縦軸がアドレス，横軸がデータ収集の繰り返し回数を表し，着色部分が前回結果から更新された範囲を表している．

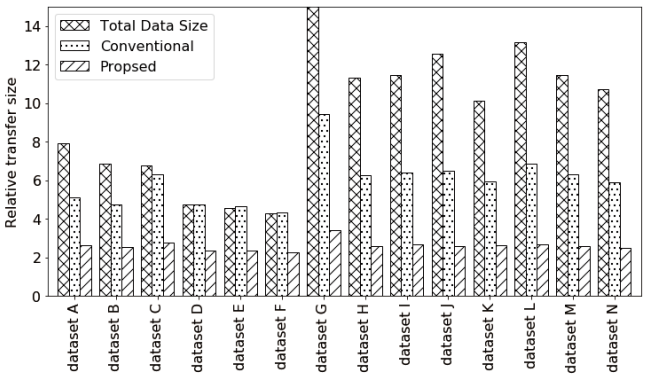

図 9：評価結果

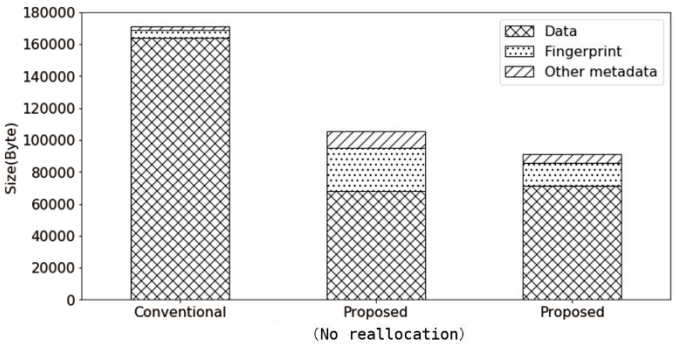

図 10：データ転送量の内訳（Dataset C）

ータルでは有利であることが確認できた．

4.3. 考察

本手法は更新箇所の空間的局所性を利用しているため，dataset B に見られたように局所性が弱い場合には既存手法よりも悪化する可能性がある．本稿で用いたデータセットは比較的局所性が高いデータであるため一定の効果は確認できたが，局所性が極端に弱いデータの場合はむしろ転送量は悪化すると考えられるため，実用上では本提案手法の効果が出ない場合は別の手法に切り替える等の対応が必要になると考えられる．

また，データ再配置による削減率は平均で 8%と高くなかった．その原因は，本評価で用いられたデータセットでは連続して更新される箇所が多くなかったことが一因として考えられる．すなわち前回更新されて集約されたチャンクのうち，次回更新されるのがそのうちの一部だけの場合は，更新されていない他のチャンクも同時に転送されてしまうことになるため，転送効率が悪くなってしまう．この点関しても今後改良の余地がある．

5. 関連研究

データ転送量を削減する技術は様々な研究が行われているが，ここでは分散データ分析での拠点間のデータ転送量を削減する手法について紹介する．

Geode[7]は拠点間に分散したデータの分析において，データ転送量削減を目指した手法である．その要素技術の１つである Subquery-Deltas は，ある分析クエリについてデータ送信元と受信先で結果をキャッシュしておき，再度同じクエリが来た場合に，前回の更新部分を検出し差分のみを転送する技術であり，差分転送方式の一方式である．文献[7]ではそれ以外にも拠点間でのタスクスケジューリング最適化や，実際の転送時間に応じて適応的に再スケジュールをすることで更なるデータ転送量削減を行っている．

SDM[8]はモバイル-クラウド間のデータ転送量を重複除去転送技術により削減した例である．本手法は，重複判定の単位がファイルベースかチャンクベースかのどちらが最適かを機械学習で識別する手法であり，チャンク割当の最適化を目指す本手法とも関連する技術である．

6. まとめ

本研究では，エッジでの計算リソース使用量を抑えながら効率的に差分を検出し，重複部分を除去して転送する技術を開発した．その結果，実際の走行車両から取得した CAN データを用いたシミュレーションでは，固定長の重複除去転送方式と比較してデータ転送量を最大で約 64%削減することができた．今後は，実機での評価を行い，データ転送量以外のオーバーヘッド，特にハッシュ値計算とデータ再配置にかかるコストを明らかにし，データ分析プラットフォームへの適用を進めていく．

文　献

[1] “Cisco Visual Networking Index: Forecast and Methodology, 2016–2021”, Cisco White paper, June 2017.

[2] “Big Data in Big Companies,” SAS Research Report, 2013

[3] Mogul, Jeffrey C., et al. "Potential benefits of delta encoding and data compression for HTTP." ACM SIGCOMM Computer Communication Review. Vol. 27. No. 4. ACM, 1997.

[4] Tridgell, Andrew, and Paul Mackerras. "The rsync algorithm.", 1996.

[5] Agrawal, Nitin, et al. "A five-year study of file-system metadata." Proceedings of the 5th USENIX conference on File and Storage Technologies. USENIX Association, 2007.

[6] Athicha Muthitacharoen, Chen Benjie, David Mazires, "A low-bandwidth network file system", ACM SIGOPS Operating Systems Review, vol. 35, pp. 174-187, 2001.

[7] Vulimiri, Ashish, et al. "Global Analytics in the Face of Bandwidth and Regulatory Constraints." NSDI. Vol. 7. No. 7.2. 2015.

[8] Widodo, Ryan NS, Hyotaek Lim, and Mohammed Atiquzzaman. "SDM: Smart deduplication for mobile cloud storage." Future Generation Computer Systems 70, pp.64-73, 2017.