小口研究室　研究紹介　(2023年度)
(お茶の水女子大学理学部情報科学科)

モバイル環境における圧縮処理を用いたIoTシステムの通信性能向上のための検討
（研究担当：伊藤　千紗）

背景/目的

さまざまなモノに通信デバイスを組み込み、インターネットに接続しようとする試みをIoT (Internet of Things) と呼ぶ.
しかしモバイル環境にあるIoTデータの収集では, 通信スループットや通信遅延を維持する必要があり, 性能向上が課題となっている. IoT通信では小規模データが大量に送信されるため, 通信環境に合わせた転送方法を検討する必要がある.

- **モバイル環境で, 効果的なデータ圧縮処理を調査し, より通信環境に合わせた転送方法を検討**
- **IoT通信でスループットを高めるのに, より効果的な送受信データ特性や圧縮手法を調査**

使用技術

SINETStream

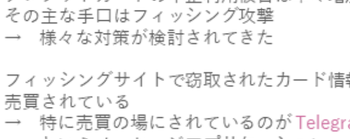
国立情報学研究所 (NII) が開発しているIoTシステム開発支援ライブラリ.
IoT通信に適したPublisher/Subscriber型の通信モデルを採用しており, 広域ネットワークを介してデータを欠損なく確実に収集・解析するための機能を提供する.

MQTT

IoT向けのTCP/IPプロトコル上に構築されたPublisher/Subscriber型のモデルに基づく軽量な通信プロトコル. わずかなコードと限られた帯域幅で, 接続されたリモートデバイスにリアルタイムの信頼性の高いメッセージングサービスを提供できる

実験概要

LTE環境でIoT通信の往復通信スループットを測定する.



実験環境

クライアント：Raspberry Pi	
機種名	Raspberry Pi 4 Computer Model B 4GB RAM
OS	Raspbian GNU / Linux 11
CPU	ARMv7 Processor rev 3 (v7l)
Main Memory	4GB
サーバ：mdx VM	
OS	Ubuntu 20.04.5 LTS
仮想CPUコア数	16
メモリ	24.19GB
仮想ディスクサイズ	100GB

実験1

送受信データ特性を変えて通信性能を比較

データサイズ	10 / 100 [KB]
データ圧縮	有 (gzip) / 無
送受信データ特性	0埋め / 乱数 / JSON

実験2

データ圧縮率を制御して圧縮による通信性能を比較

データサイズ	10 / 100 [KB]
データ圧縮	有(gzip) / 有(zstd) / 無
圧縮率	0.1 / 0.2 / 0.3 / 0.4 / 0.5 / 0.6 / 0.7 / 0.8 / 0.9 / 1.0

実験結果

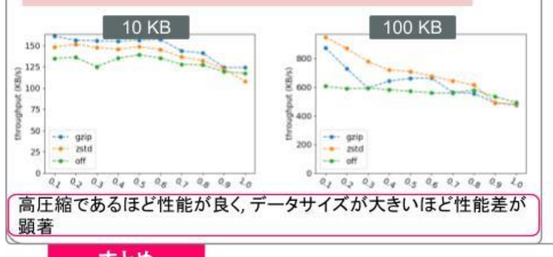
※JSONのデータサイズは厳密には2.8KB,11KB,112KBであるが, ここでは近似して比較を行う

※1000回の累積処理時間

実験1：データ特性の違いによる通信スループットの比較

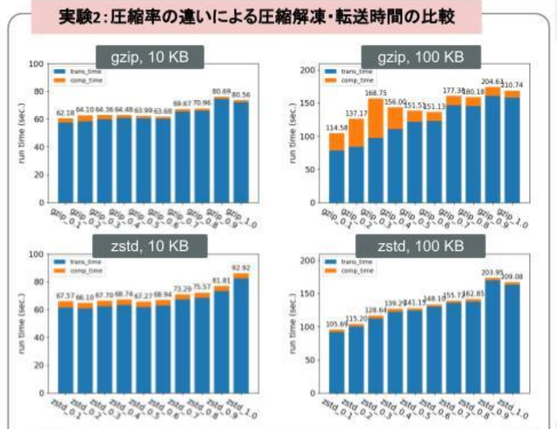
乱数データでは, データ圧縮の効果がみられない
0埋めデータでは, 大幅な性能向上がみられ, データサイズが大きくなるほど性能差が大きく, JSONデータでは, 小さいデータサイズの場合に高性能
➢ 送受信するデータの性質を考慮した圧縮処理の適用が重要

実験2：圧縮率の違いによる通信スループットの比較



高圧縮であるほど性能が良く, データサイズが大きいほど性能差が顕著

実験2：圧縮率の違いによる圧縮解凍・転送時間の比較



gzipとzstdのいずれも全体の実行時間に対して圧縮解凍時間はかなり短く, 圧縮率との相関はみられない
➢ 圧縮時間が全体の実行時間に与える影響は小さいため, 実行時間の観点においても圧縮処理が有効
gzipではデータサイズが大きくなると圧縮時間が長くなる
zstdではデータサイズに関係なく圧縮時間はほぼ一定
➢ データサイズに応じてアルゴリズムを適切に選択することが重要

まとめ

IoT通信でスループットを高めるのに, より効果的な送受信データ特性や圧縮手法を調査し, その結果, 送受信するデータサイズやデータ特性に応じた圧縮処理の適用が性能向上に効果的であること, 実行時間の観点においても圧縮処理が有効であることがわかった. 今後は引き続き通信環境に合わせてIoT通信の転送効率を高める手法を検討する.

SNSにおけるサイバー犯罪対策: 交グラフを活用した犯罪グループ探索アルゴリズム
（研究担当：大原望乃）

研究背景　フィッシング攻撃とTelegram

・クレジットカードの不正利用被害は年々増加
・その主な手口はフィッシング攻撃
　→　様々な対策が検討されてきた

・フィッシングサイトで窃取されたカード情報がSNS上で売買されている
　→　特に売買の場にされているのがTelegram（テレグラム）というメッセージアプリケーション

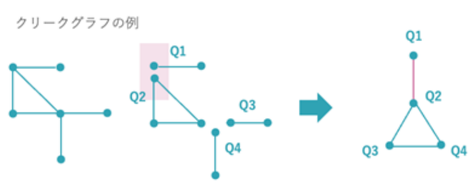
先行研究と課題　モニタリング / 交グラフの活用

・検索APIを使い，カード情報売買に使われる特徴的な文字列を検知可能なモニタリングツールを整備

・結果はカード会社に連携し，利用確認・利用停止などで対応（2022 日本総合研究所）

・犯罪に関与するグループをもっと効率よく見つけたい
　→　犯罪コミュニティの中心を特定し、そこから辿る

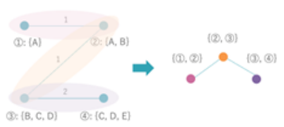
　→　交グラフを用いた犯罪グループ探索手法を提案



提案手法　交グラフとグラフの彩色

Step 1. データベースからユーザリストをダウンロード
Step 2. ダウンロードしたデータのマイニング
Step 3. 複雑ネットワークのグラフを生成
Step 4. エッジの重み付け

Step 5. 交グラフに変換

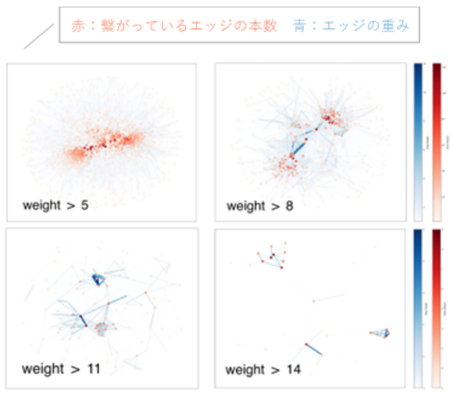


Step 6. 交グラフからコミュニティを抽出
　→　クラスタリングを行い，クラスタごとにノードを彩色
　→　元のグラフに戻した時，複数のクラスタに所属するノード（紫）はコミュニティの中心である可能性が高い

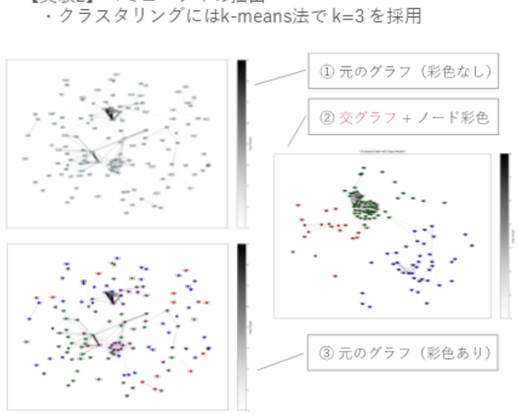
・また，そのままだとデータ数が多すぎるため，重要度の低いと思われるノードとエッジを削減する処理が必要

実験　閾値によるノード・エッジ削減の比較 / コミュニティの抽出

【実験1】閾値によるノード・エッジ削減の比較



【実験2】コミュニティの抽出
・クラスタリングにはk-means法で k=3 を採用



実験1：他のグループ群と所属するグループが被りやすいグループ群のノードが赤く可視化
実験2：交グラフで複数のクラスタに属するグループ群のノードが紫で彩色
➡ 多くのノードが一致 推論のI妥当性が高まる

今後の展望

・クラスタリング手法の比較検討　・大規模データへの対応　等

準同型暗号を用いた比較演算におけるエンコード手法による性能比較
（研究担当：内藤　華）

研究背景

◆ 個人情報などの電子データが第三者のクラウド上でデータ分析に利用されており、情報漏洩や改ざんを防ぐために暗号化が必要
◆ **準同型暗号**を用いると暗号化状態での計算が可能であり、第三者の目に触れることなく処理結果を取り出すことができる
◆ データを分析に利用する際にはエンコードが必要

準同型暗号で暗号化したデータに対して**比較演算**を行い、**エンコード手法**による処理性能の比較を行う

エンコード手法

◆ バイナリ・エンコーディング
- 整数データを二進数で表現
- ビット長はデータの範囲の大きさに従って対数的に増加

◆ ビットマスク・エンコーディング
- 整数kを表現する際はビット列の最初のkビットを1とする
- ビット長はデータ範囲の大きさに従って線形に増加

Age	Age in Binary	Age in Bitmask
53	110101	1…10…0 (53, 22)
40	101000	1…10…0 (40, 35)
25	011001	1…10…0 (25, 50)
29	011101	1…10…0 (29, 46)
33	100001	1…10…0 (33, 42)

比較演算におけるビットマスク・エンコーディングの利点

1回のAND/OR演算で計算可能！

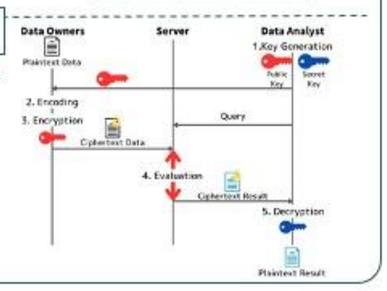
MIN(4, 7)	MAX(4, 7)
4 = 1111000000	4 = 1111000000
AND 7 = 1111111000	OR 7 = 1111111000
Ans 4 = 1111000000	Ans 7 = 1111111000

実験概要

2種類の手法でエンコードした右図のテーブルに対してMAX演算、フィルター演算を行い、処理時間を計測した
◆ MAX演算
テーブルから年齢の最大値を求める
◆ フィルター演算
貯金が“quite rich”以上のレコード抽出

Age	Gender	Savings	Credit amount
22	female	little	5951
49	male	little	2096
53	male	quite rich	2835
…			
32	female	moderate	1282

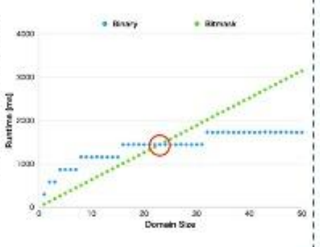
データ暗号化、演算、復号の流れ



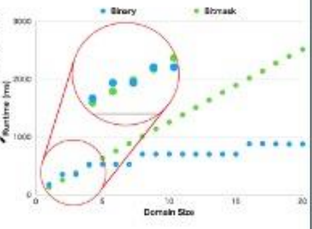
処理の手順
① データ解析者が公開鍵/秘密鍵を生成しデータ所有者に公開鍵を送信、データ所有者がデータをエンコードした後に暗号化
② データ所有者が暗号データを、データ解析者がクエリをサーバに送信し、サーバが演算を行う
③ サーバが演算結果をデータ解析者に送信し、データ解析者が秘密鍵で復号

実験結果

MAX演算
- ビットマスクの方がビットごとの演算は高速だがビット長が大きい
- データの範囲が23未満ではビットマスク、23以上ではバイナリが速い



フィルター演算
- ビットマスクの方がビットごとの演算はわずかに速い
- データの範囲が5未満ではビットマスク、5以上ではバイナリが速い



https://www.is.ocha.ac.jp/~oguchi_lab/　　oguchi@is.ocha.ac.jp