ROSCon JP 2024
ロボットソフトウェア開発における MCAP活用

株式会社サイバーエージェント AI Lab
Activity Understanding Team
吉村 康弘

CyberAgent AI Lab

自己紹介

吉村 康弘 | Yasuhiro Yoshimura

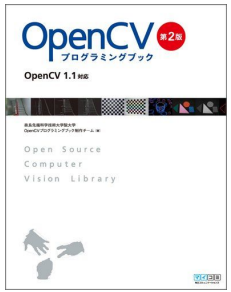
株式会社サイバーエージェント AI Lab

Activity Understanding Team

Research Engineer

コンピュータビジョン、ロボティクスの研究に従事

OpenCVコントリビューター



本発表の狙い

- **MCAPの理解を深める**
- **MCAP形式での効率的なrosbag2記録方法を理解する**
- **既存のrosbag、rosbag2資産からの移行方法、注意点を理解する**

※本発表ではROS 2 Humble時点での機能を紹介します。

本発表の狙い

- MCAPの理解を深める
- MCAP形式での効率的なrosbag2記録方法を理解する
- 既存のrosbag、rosbag2資産からの移行方法、注意点を理解する

※本発表ではROS 2 Humble時点での機能を紹介します。

端的に言うと「これからは MCAP使おう！」

会場の皆様への質問（1/2）

MCAPについて、名前は聞いたことある、知っている方は挙手ください！

会場の皆様への質問（2/2）

ROS 2を使ったロボット開発でMCAP使ったことある方は挙手ください！

はじめに

- **ロボットソフトウェア開発におけるデータ記録の重要性**
  - センサデータ、実行時の状態、ログなどをデータ記録しておくことでオフラインでアルゴリズム検討、デバッグができる
    - ROS 1/ROS 2を使った開発ではrosbag/rosbag2形式で記録することが多い
  - 記録時にできるだけ取りこぼしがなく、再生時には高速にデータが読み込めて、相互運用しやすいことが望ましい

MCAPとは

ROS 2 Iron IrwiniからrosbagのデフォルトストレージにMCAPが採用されたフォーマット。

- https://mcap.dev/guides
- https://foxglove.dev/blog/mcap-as-the-ros2-default-bag-format

詳細な仕様に興味のある方は https://mcap.dev/spec を参照ください。

MCAPとは

https://mcap.dev/ で言及されているように以下のメリットがある。

- 書き込みのスループットが高い
- 任意のタイミングへのシークが高速
- 破損データの修復が容易
- スキーマ定義が埋め込まれているため、パッケージ依存が少ない
- 様々なプログラミング言語でMCAPファイルを読み書きできる
  - C++、Go、Python、Rust、Swift、TypeScript

記録

- **インストール**
  - ROS 2 Humble時点でrosbag2のMCAPストレージはデフォルトではないのでパッケージのインストールが必要

```
sudo apt install ros-humble-rosbag2-storage-mcap
```

記録

- **基本的な書き込み例**
  - ストレージはMCAP、全てのトピックを記録する例。
  - **-s**オプションでストレージとしてMCAPを指定。
  - 詳細な方法についてはAppendix参照。

```
ros2 bag record -s mcap --all
```

データ変換

既存のROS 1のrosbag、ROS 2のrosbag2（SQLite3）資産からROS 2のrosbag2（MCAP）へのデータ変換方法はいくつかあります。ここでは以下の2つを紹介。

- **ros2 bag convert**
- **MCAP CLI**

※2024/9/25時点で、rosbags（Pythonパッケージ）はROS 1のrosbagからROS 2のrosbag2（MCAP）に変換できない点に注意が必要。詳細はAppendix参照。

データ変換

ros2 bag convertで変換

- **ros2 bag convert**コマンドで変換できる

```
ros2 bag convert -i <input_bag> storage_id -o <output_option_yaml>
```

- **オプション記述例**

```
output_bags
- uri: output_mcap
  storage_id: mcap
```

- 詳細は https://github.com/ros2/rosbag2/blob/humble/README.md#converting-bags 参照

データ変換

MCAP CLIで変換 ※MCAP CLIの詳細についてはAppendix参照

- **bag(ROS 1) -> MCAP**

```
mcap convert <input_bag> <output_mcap>
```

- **bag2(ROS 2、SQLite3) ->MCAP**

```
mcap convert <input_db3> <output_mcap>
```

ベンチマーク

以下の観点でベンチマークを実施。

- **書き込みスループット**
  - この値が高い方が大量のデータを書き込んだ時に取りこぼしにくい

※ベースにしているベンチマークコード（下記URL参照）がROS 2 Iron以降を想定しているため、ROS 2 Jazzy上で実施

https://github.com/james-rms/rosbag2/tree/plugin-comparison/rosbag2_performance/rosbag2_storage_plugin_comparison

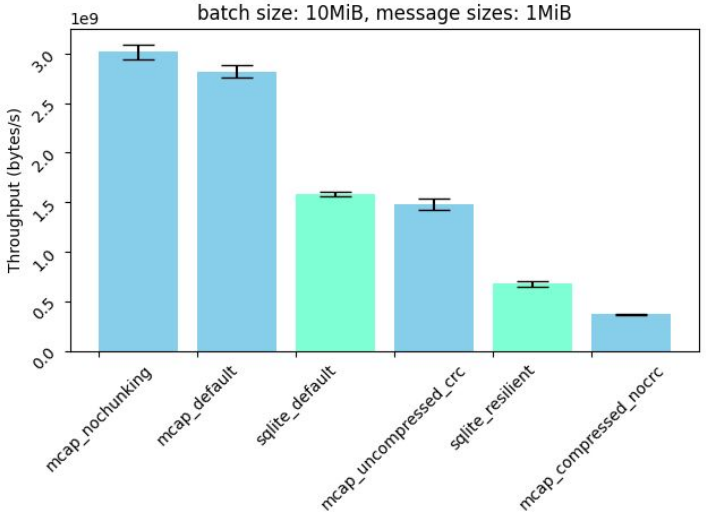
ベンチマーク

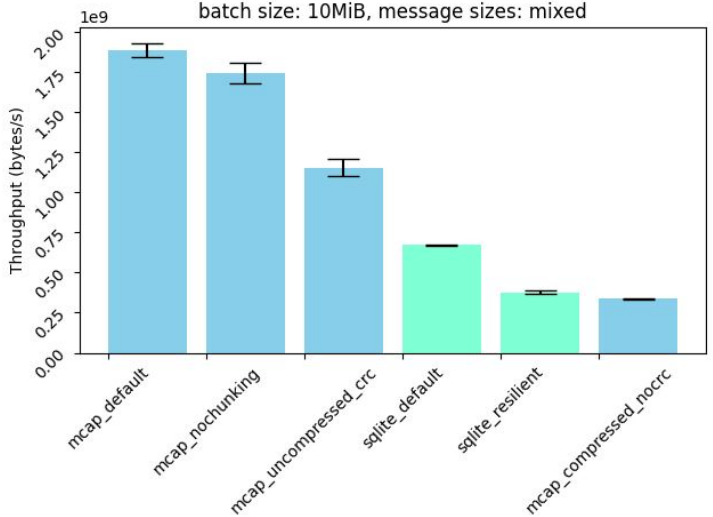
- **書き込みスループット**
  - https://mcap.dev/guides/benchmarks/rosbag2-storage-plugins を追試
  - 今回計測に用いたプリセットは以下の通り。詳細はAppendix参照。

プリセット名	概要
sqlite_default	SQLite3 storage pluginのデフォルト構成
sqlite_resilient	修復のための構成（SQLite3 ）
mcap_default	MCAP storage pluginのデフォルト構成
mcap_nochunking	チャンクなしでスループット優先にした構成
mcap_uncompressed_crc	修復のためのCRC計算（MCAP）
mcap_compressed_nocrc	mcap_defaultに対してチャンク毎の圧縮

ベンチマーク

- **書き込みスループット**
  - 多くのケースでSQLite3よりもMCAPの方がスループットが高い
  - 下記ケースでは、MCAP default、チャンクなしのスループットが高い





ユースケース別rosbag2活用例

- **破損が見つかったら修復したい**
  - **CRC（Cyclic Redundancy Check）を付与しておく**
    - https://github.com/ros2/rosbag2/blob/humble/rosbag2_storage_mcap/README.md#writer-configuration

Field	Type / Values	Description
noChunkCRC	bool	Disable CRC calculation for Chunks.
noAttachmentCRC	bool	Disable CRC calculation for Attachments.
enableDataCRC	bool	Enables CRC calculation for the entire Data section.
noSummaryCRC	bool	Disable CRC calculation for the Summary section.

  - ただし、CRC計算を有効にすると、rosbag2の書き込みスループットが落ちるため注意

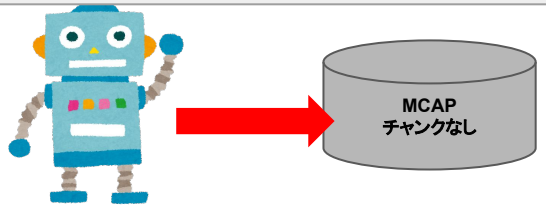
ユースケース別rosbag2活用例

- **書き込みスループット優先で書き込みたい（1/2）**
  - 書き込みのオーバーヘッドをなくすため、チャンクなしで記録する

```
ros2 bag record -s mcap --all --storage-preset-profile fastwrite
```

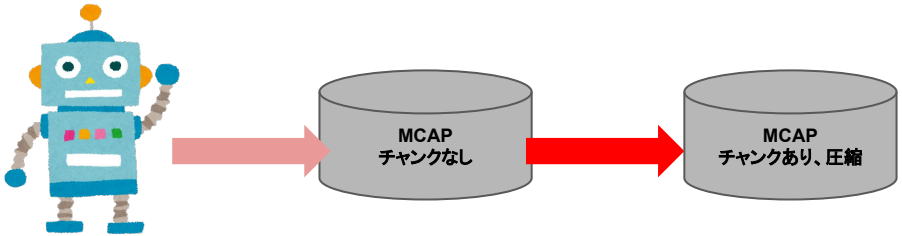
    - プリセット**fastwrite**は以下の設定に相当

```
noChunking: true
noSummaryCRC: true
```



ユースケース別rosbag2活用例

- **書き込みスループット優先で書き込みたい（2/2）**
  - 効率的に読み込むために後からメッセージインデックスを付与したり、データサイズ削減のため圧縮
    - **ros2 bag convert（storage_preset_profile=zstd_small）**で後処理として変換



ユースケース別rosbag2活用例

- **アップロードしやすいサイズに分割したい**

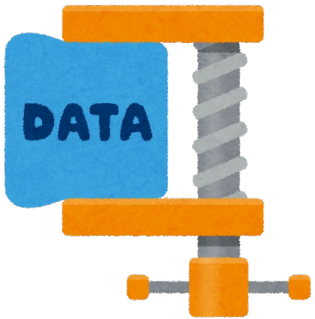
  オンラインストレージで保管するため、アップロードしやすく記録したい。

  - **分割例１：タイムスタンプでファイル分割**
    - 一定の時間間隔でデータを分割して記録
    - 不要なデータを選択して削除しやすい

  - **分割例２：まとまったトピック単位でファイル分割**
    - 例えば、以下のような分類で別々のMCAPとして記録
      - 取りこぼしたくないトピック、取りこぼしてもいいトピック
      - サイズが小さいトピック、大きいトピック
    - 必要に応じて、**mcap merge**コマンド等でタイムスタンプ順に統合

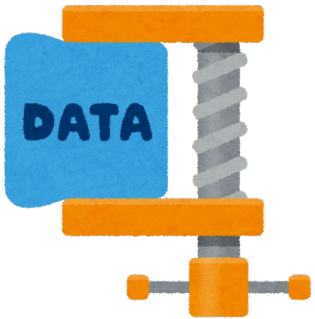
ユースケース別rosbag2活用例

- **データサイズを小さくしたい**
  - 圧縮することでファイルサイズを削減できる
  - ただし、圧縮処理によってCPU、メモリ使用量が増加するためトレードオフがある点に注意



ユースケース別rosbag2活用例

- データサイズを小さくしたい
  - **Zstd（compressionLevel=Slowest）で圧縮する**
    - ただし、処理が重くなる可能性があるので後処理として圧縮するのがよい
    - **ros2 bag convert（storage_preset_profile=zstd_small）**で変換



ユースケース別rosbag2活用例

- データサイズを小さくしたい
  - 静止画圧縮、動画像圧縮
    - CompressedImage
      - WebP、JPEG、PNG圧縮された静止画データを記録できる
      - https://docs.foxglove.dev/docs/visualization/message-schemas/compressed-image/
    - CompressedVideo
      - H.264でエンコードされたビデオ ストリームを記録できる
      - https://docs.foxglove.dev/docs/visualization/message-schemas/compressed-video/
      - https://foxglove.dev/blog/announcing-h264-support-in-foxglove

ユースケース別rosbag2活用例

- **ROS 2を使わないプロジェクトでも使いたい**
  - ROS 1、ROS 2に依存しないスキーマを使って書き出しておくと、ROS 2に依存しないプロジェクトでも活用できる。
  - このあたりのパッケージを使うとよい。
    - https://pypi.org/project/foxglove-schemas-protobuf/
    - https://pypi.org/project/foxglove-schemas-flatbuffer/

ユースケース別rosbag2活用例

- **ROS 2を使わないプロジェクトでも使いたい**
  - 独自スキーマを用いて記録もできる。以下に例を示す。

スキーマ定義例

```
syntax = "proto3";

message Trajectory2d {
   message Point2d {
     float x = 1;
     float y = 2;
   }
   repeated Point2d points = 1;
 }
```

https://lorenzopeppoloni.com/posts/nestedmessagepy/

MCAP書き込み例

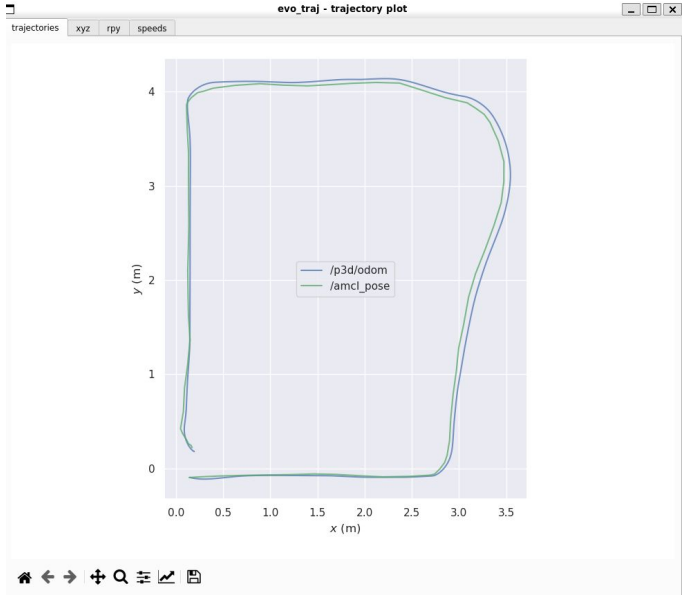
```
import time
from mcap_protobuf.writer import Writer
from trajectory2d_pb2 import Trajectory2d

def write_trajectory_frame(writer: Writer, now: int):
    trajectory  = Trajectory2d()
    trajectory.points.add(x=10, y=30)
    trajectory.points.add(x=14, y=22)
    writer.write_message(
        topic="/trajectory",
        log_time=now,
        message=trajectory,
        publish_time=now,
    )

if __name__ == "__main__":
    with open("output.mcap", "wb") as f, Writer(f) as writer:
        now = time.time_ns()
        write_trajectory_frame(writer, now)
```

ユースケース別rosbag2活用例

- **evoを使いたい**
  - SLAM評価ツールevo https://github.com/MichaelGrupp/evo はMCAP対応している！



まとめ

- MCAPはROS 2 Iron IrwiniからRosbag2のデフォルトストレージに採用されたフォーマット。
- MCAPは以下のような特長がある
  - 書き込みのスループットが高い
  - 任意のタイミングへのシークが高速
  - 破損データの修復が容易
- よく使われるツール、過去資産（ROS 1/ROS 2 rosbag）からの移行方法も整っている
- **これからはMCAPをぜひ使いましょう！**

Reference

- https://mcap.dev/
- https://mcap.dev/guides
- https://mcap.dev/spec
- https://mcap.dev/guides/cli
- https://mcap.dev/files/evaluation.pdf

Reference

- https://foxglove.dev/blog/announcing-the-mcap-storage-plugin-for-ros2
- http://download.ros.org/downloads/roscon/2022/MCAP%20A%20Next-Generation%20File%20Format%20for%20ROS%20Recording.pdf
- https://foxglove.dev/blog/best-practices-for-processing-and-analyzing-robotics-data
- https://foxglove.dev/blog/best-practices-for-recording-and-uploading-robotics-data

Appendix：記録

- **オプション指定による書き込み例**
  - **--compression-mode**オプションでモードを指定
  - **--compression-format**で圧縮フォーマットを指定
  - 詳細は https://github.com/ros2/rosbag2/blob/humble/README.md#recording-with-compression 参照

```
ros2 bag record -s mcap --all --compression-mode file
--compression-format zstd
```

Appendix：記録

- **コンフィグファイルを用いる書き込み例**
  - **--storage-config-file**オプションでコンフィグファイルを指定
  - コンフィグファイル記述方法は次ページに記載

```
ros2 bag record -s mcap --all --storage-config-file <storage_config_yaml>
```

Appendix：記録

- コンフィグファイルの記述例
  - 詳細は https://github.com/ros2/rosbag2/blob/humble/rosbag2_storage_mcap/README.md 参照

```
noChunkCRC: false
noChunking: false
noMessageIndex: false
noSummary: false
chunkSize: 786432
compression: "Zstd"
compressionLevel: "Fast"
forceCompression: false
```

Appendix：記録

- **ros2 bag recordのプリセットプロファイル**
  - よく使いそうな設定の組み合わせのプリセットプロファイルが提供されている
    - fastwrite：書き込みスループット優先
    - zstd_fast：Zstd圧縮、速度優先
    - zstd_small：Zstd圧縮、サイズ優先
  - 詳細は https://github.com/ros2/rosbag2/tree/humble/rosbag2_storage_mcap#storage-preset-profiles 参照

Appendix：記録

- ros2 bag recordのプリセットプロファイル
  - fastwrite：書き込みスループット優先

```
noChunking: true
noSummaryCRC: true
```

  - zstd_fast：Zstd圧縮、速度優先

```
compression: "Zstd"
compressionLevel: "Fastest"
noChunkCRC: true
```

  - zstd_small：Zstd圧縮、サイズ優先

```
compression: "Zstd"
compressionLevel: "Slowest"
chunkSize: 4194304
```

Appendix：データ変換

rosbags

- https://ternaris.gitlab.io/rosbags/index.html
- インストールも簡単

```
pip install rosbags
```

- 基本的な使い方

```
rosbags-convert --src <input_bag> --dst <output_folder>
```

  - ただし、この方法で変換すると2024/9/25時点ではSQLite3形式となる

Appendix：MCAP CLI

- **MCAP CLIインストール**
  - https://github.com/foxglove/mcap/releases で配布されている
  - Linux、x86_64の場合のインストール例は以下の通り。

```
wget https://github.com/foxglove/mcap/releases/download/releases%2Fmcap-cli%2Fv0.0.47/mcap-linux-amd64
chmod +x mcap-linux-amd64
sudo mv mcap-linux-amd64 /usr/bin/mcap
```

Appendix：MCAP CLI

- **info: MCAPファイルの情報表示**

```
mcap info <input_mcap>
```

```
$ mcap info rosbag2_2020_09_23-15_58_07.mcap
library:   mcap go v1.4.1
profile:   ros2
messages:  1535
duration:  1m16.854162986s
start:     2020-09-23T22:58:07.524769241Z (1600901887.524769241)
end:       2020-09-23T22:59:24.378932227Z (1600901964.378932227)
compression:
        zstd: [81/81 chunks] [679.38 MiB/323.69 MiB (52.36%)] [4.21 MiB/sec]
channels:
        (1)  /rosout                                0 msgs            : rcl_interfaces/msg/Log [ros2msg]
        (2)  /lidar_front/points_raw              768 msgs (9.99 Hz)   : sensor_msgs/msg/PointCloud2 [ros2msg]
        (3)  /parameter_events                      0 msgs            :
《中略》
attachments: 0
metadata: 0
```

Appendix：MCAP CLI

- **merge: タイムスタンプに基づいた複数MCAPファイルのマージ**

```
mcap merge <input1_mcap> <input2_mcap> --compression "zstd" -o <output_mcap>
```

- **compress: 圧縮**

```
mcap compress <input_mcap> --compression "zstd" -o <output_mcap>
```

- **decompress: 展開**

```
mcap decompress <input_mcap> -o <output_mcap>
```

Appendix：MCAP CLI

- doctor: MCAPファイルのチェック

```
mcap doctor <input_mcap>
```

- recover: 破損データの修復

```
mcap recover <input_mcap>
```

Appendix：ベンチマーク

- **書き込みスループット**
  - https://mcap.dev/guides/benchmarks/rosbag2-storage-plugins を追試
  - 以下のようなパターンで計測。
    - All 1MiB
    - All 10KiB
    - All 100B
    - Mixed：以下の割合で混ぜたもの
      - 1MiB：70%
      - 10KiB：20%
      - 100B：10%
  - 以降、いくつかのケースをピックアップ。詳細結果はAppendix参照。

Appendix：ベンチマークで使用したプリセット

- **sqlite_default**
  - SQLite3 storage pluginのデフォルト構成。
  - 記録時に中断された場合、破損する可能性がある。

- **sqlite_resilient**
  - SQLite3 storage pluginで破損に対してある程度回復できるようにした設定
  - パフォーマンスが犠牲になるがファイル全体が破損するリスクは減らせる

- **mcap_default**
  - MCAP storage pluginのデフォルト構成。
  - インデックスを備えた非圧縮のチャンクありMCAP

https://github.com/james-rms/rosbag2/blob/plugin-comparison/rosbag2_performance/rosbag2_storage_plugin_comparison/scripts/sweep.py

Appendix：ベンチマークで使用したプリセット

- **mcap_nochunking**
  - チャンクを生成しないのでスループット優先のプリセット
  - 効率的に読み取るために後でインデックスを再作成する必要あり

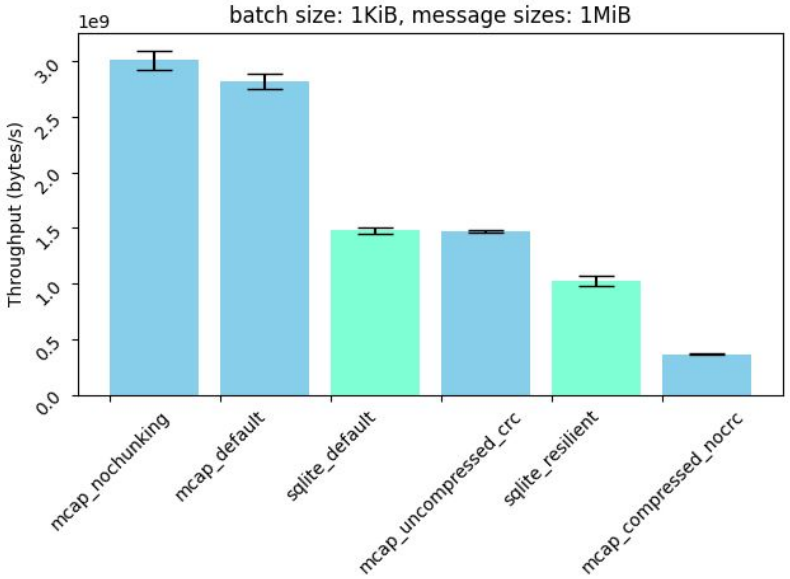
- **mcap_uncompressed_crc**
  - 基本はmcap_defaultと同じだがチャンクに破損したデータが含まれているかどうかを識別するためCRCを計算

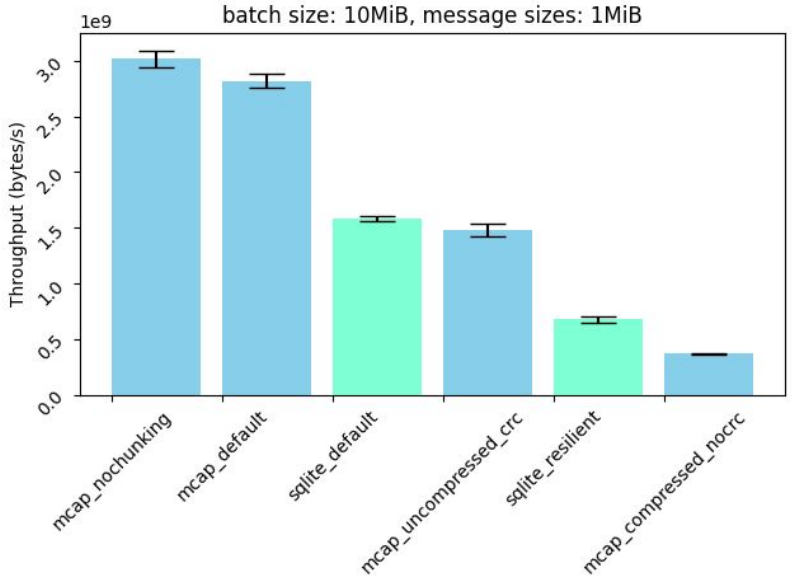
- **mcap_compressed_nocrc**
  - 基本はmcap_defaultと同じだがチャンクごとに圧縮する

https://github.com/james-rms/rosbag2/blob/plugin-comparison/rosbag2_performance/rosbag2_storage_plugin_comparison/scripts/sweep.py

Appendix：ベンチマーク

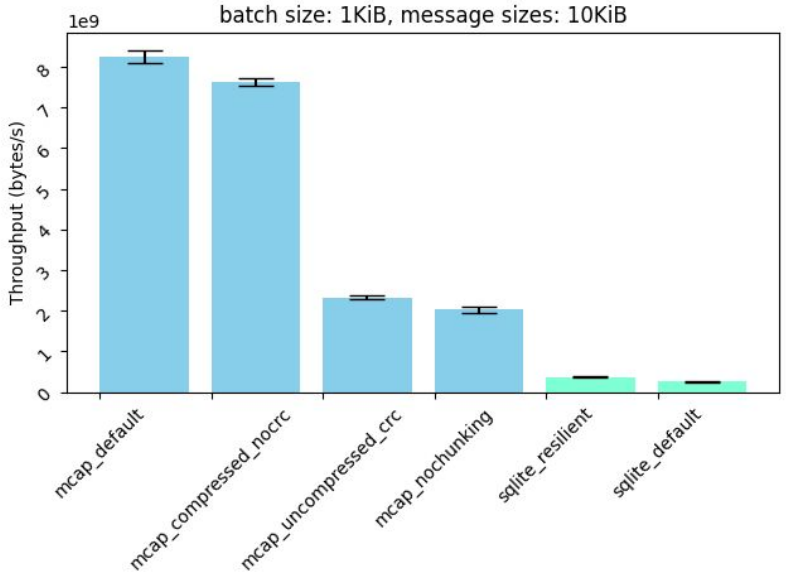
- 書き込みスループット

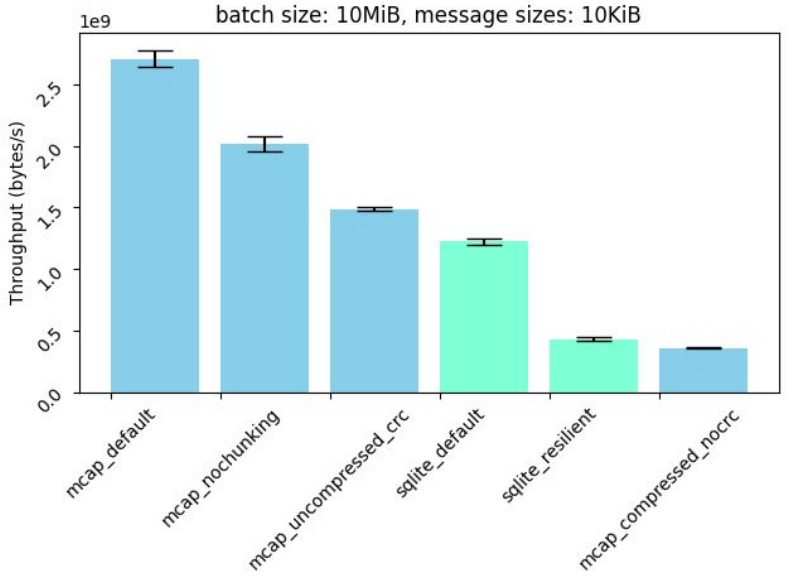




Appendix：ベンチマーク

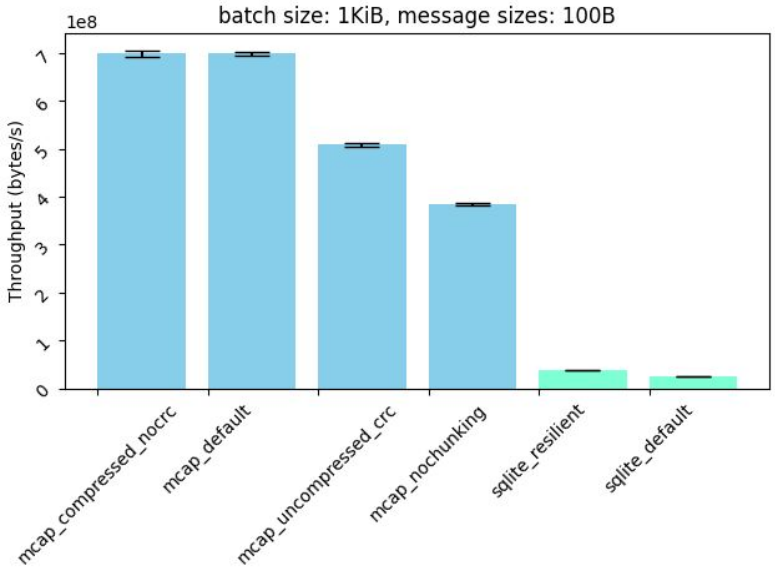
- 書き込みスループット

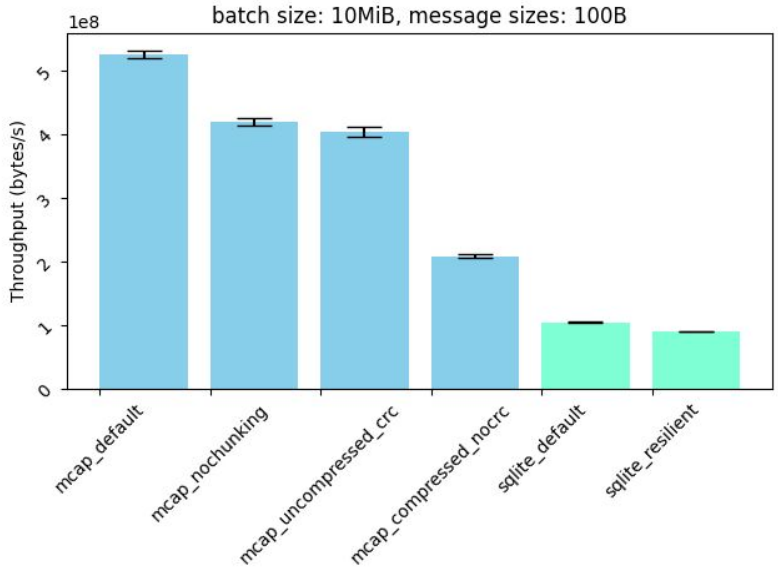




Appendix：ベンチマーク

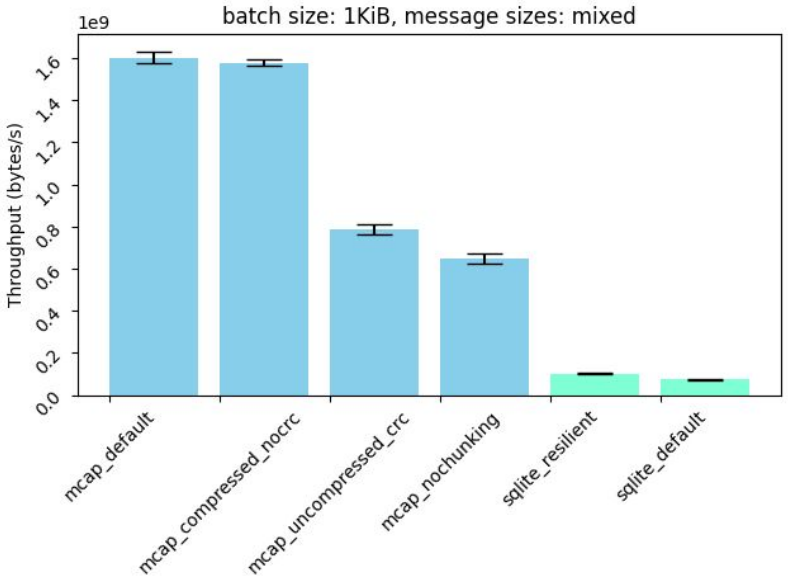
- 書き込みスループット

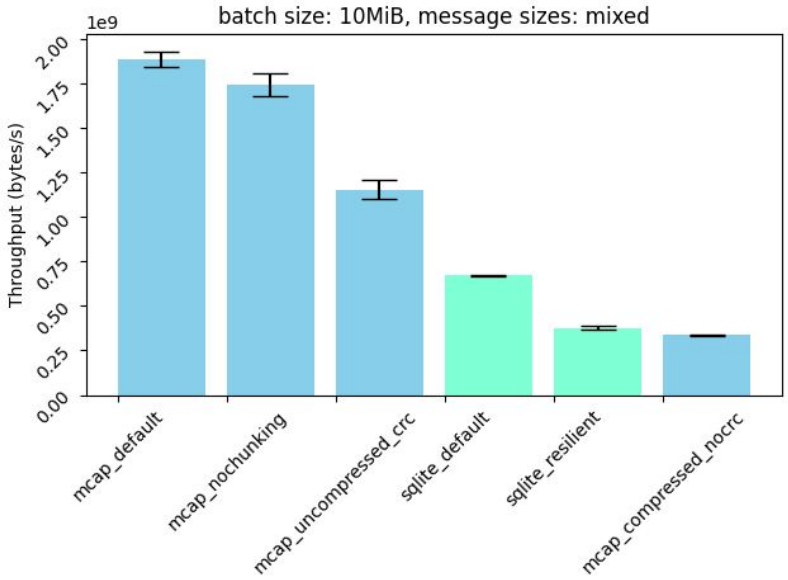




Appendix：ベンチマーク

- 書き込みスループット





Appendix：ユースケース別rosbag2活用例

- **データサイズを小さくしたい**
  - **Chunk Compression**
    - チャンク単位で圧縮してデータサイズが減らすことで、ディスクI/Oが減る
    - 圧縮形式はZstd、LZ4をサポート
    - 閾値を超えるまでディスクに書き込まれないので、クラッシュなどで記録が中断されるとメモリに保持していたデータは記録されない
      - この挙動が気になる場合はチャンクサイズを小さくする必要あり
    - 詳細は https://foxglove.dev/blog/understanding-mcap-chunk-size-and-compression 参照

Appendix：ユースケース別rosbag2活用例

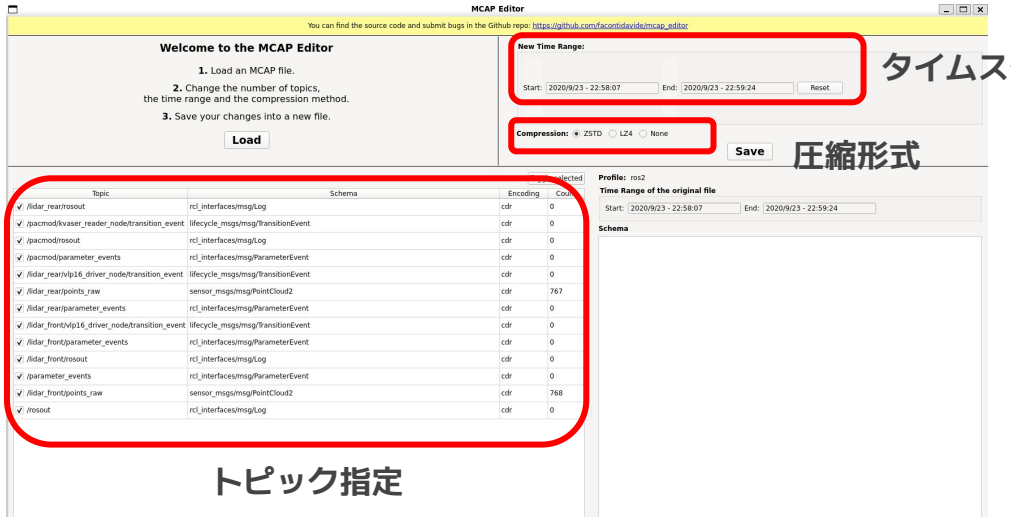
- **rqt_bagを使いたい**
  - **ros2 run rqt_bag rqt_bag <input_mcap>**で以下のエラーが出る。

```
sqlite3.DatabaseError: file is not a database
```

  - 2024/9/25時点でROS 2 HumbleのRqt_bagはMCAP非対応のため要注意。
    - ROS 2 JazzyのRqt_bagはMCAP対応している
  - **ROS 2 Humbleでも使いたい人はバックポートのPR出しましょう！**

Appendix：MCAP編集ツール

- MCAP editor
  - https://github.com/facontidavide/mcap_editor
  - タイムスタンプのレンジ指定、圧縮形式変更、トピック削除がGUIでできる



Appendix：MCAP編集ツール

- **Kappe**
  - https://pypi.org/project/kappe/
  - CUIベースの編集ツール
  - 以下の機能がある
    - トピックのリネーム、削除
    - ROS header time更新
    - Tf削除
    - 特定時間帯のみのトリミング
    - etc...
  - インストールも簡単

```
pip install kappe
```

コンフィグファイル例（トピック名のリネーム）

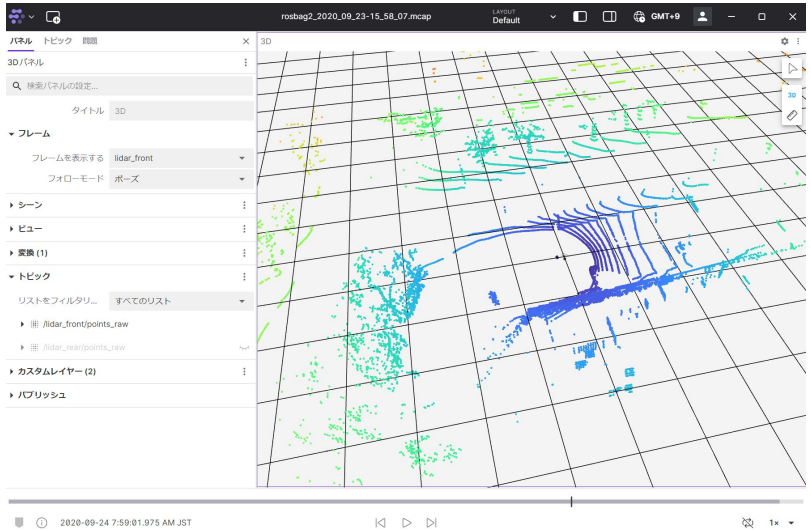
```
topic:
  mapping:
    /points: /sensor/points
```

変換コマンド

```
kappe convert --config config.yaml input.mcap
```

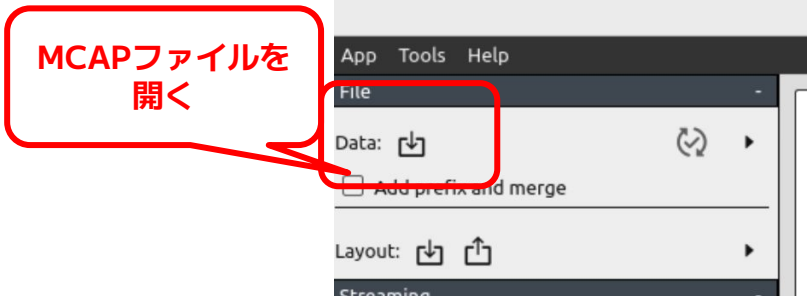
Appendix：MCAP可視化ツール

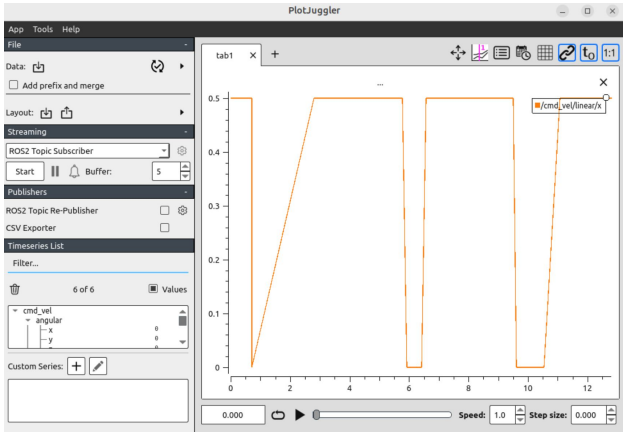
- **Foxglove Studio** https://foxglove.dev/
  - Foxglove Technologies, Inc.が開発しているデータ可視化ツール
  - ROS 1 rosbag、ROS 2 rosbag2（SQLite3、MCAP）対応



Appendix：MCAP可視化ツール

- **PlotJuggler** https://plotjuggler.io/
  - ROS 1、ROS 2サポートしたプロットツール
  - rosbag2（SQLite3、MCAP）サポート
    - https://foxglove.dev/blog/plotjuggler-adds-support-for-mcap





Appendix：様々なプログラミング言語でMCAPを読み書きする

- APIドキュメントは https://mcap.dev/reference 参照
- 言語によって、サポートしている機能が異なる点に注意
  - https://mcap.dev/reference#feature-matrix 参照