

pure Ruby Apache Arrow実装の 活用方法

須藤功平

株式会社クリアコード

Ruby Association Activity Report
2026-08-04

Rubyアソシエーション開発助成金制度

Rubyアソシエーションは、Ruby処理系、ライブラリ及びフレームワーク等に関連した...開発プロジェクトを募集し、助成金を交付しています。

[「<https://www.ruby.or.jp/ja/news/20250819>」より引用]

応募内容

pure Ruby Apache Arrow実装の開発

本プロジェクトでは、低コストでのデータ交換のみにフォーカスしたpure RubyのApache Arrowシリアライザー・デシリアライザーを実装する。pure Rubyで実装することにより、より手軽にRubyでApache Arrowを扱えるようにし、データ処理関連のRubyライブラリーが増える下地を整えることを狙う。

[「<https://www.ruby.or.jp/ja/news/20251030>」より引用]

開発助成金制度のメリット

- ✓ お金をもらえる（75万円）
- ✓ メンターがサポートしてくれる
- ✓ 宣伝の機会が増える（この場とか）

これはうれしい！と思った人は今年度の募集に応募しよう！

私にとってのメリット

- ✓ 何年もやった方がいいよなぁと思っていたアイデアを実現できた

このプロジェクトでのチャレンジ

- ✓ Apache Arrow・Ruby開発者を増やす
 - ✓ 興味がある人は一緒にやろうぜ！と告知
 - ✓ ↑向けに関連情報を整理
 - [実装者向けのApache Arrowフォーマットの説明](#)
 - ✓ RubyKaigi 2026でRuby改良アイデアを共有
- ✓ 開発助成金制度応募者を増やす
 - ✓ メンターとのやりとり他を公開場所を実施
 - <https://app.element.io/#/room/#red-data-tools/ruby-association-grant-2025:matrix.org>

チャレンジ結果

- ✓ Apache Arrow開発者
 - ✓ 1人増えた ([@hiroyuki-sato](#))
- ✓ Ruby開発者 (進め方とか設計とかをサポート (?) している)
 - ✓ [@hasumikin](#) : 文字列にビット演算を追加
 - ✓ [@himura467](#) : IO::Bufferの改良
 - ✓ 文字列の内部表現に\0終端を要求しないやつも↑の延長
- ✓ 開発助成金制度応募者
 - ✓ 応募したくなかった人はあとで教えてね

プロジェクトの成果

- ✓ pure Ruby FlatBuffers実装（！）
Apache Arrowフォーマットは一部FlatBuffersを使っている
 - ✓ なかったんだもん！
 - ✓ google/flatbuffersにC++実装もPRしたけど
反応がなかったから諦めた
<https://github.com/google/flatbuffers/pull/8800>
 - ✓ プロジェクトの途中で[lutaml/unibuf](#)が登場
FlatBuffersだけでなくProtocol Buffersとかにも対応
でも、必要な機能が未実装で使えなかった
- ✓ pure Ruby Apache Arrow実装

Apache Arrow

- ✓ メモリー上の大量データ用の
高速なカラムナーフォーマット
- ✓ メモリー上の大量データ用の
高速なデータ処理ライブラリー
- ✓ ...

pure Ruby Apache Arrow実装

- ✓ メモリー上の大量データ用の
高速なカラムナーフォーマット
- ✓ メモリー上の大量データ用の
高速なデータ処理ライブラリー
- ✓ ...

なぜpure Rubyなのか

Apache Arrow ユーザーを 増やしたい

Apache Arrowユーザーを増やす

- ✓ インストールの障害を減らす
- ✓ ユーザーになると嬉しいことを増やす

拡張ライブラリーとの比較

✓ 小さなインストールサイズ

- ✓ pure Ruby: 42KB

- ✓ polars-df: 119MB

✓ インストールに失敗しない

- ✓ ビルドエラーがない

メモ: precompiled gemのことを考えている人がいないかをチェックすること

Apache Arrowユーザー

- ✓ Apache Arrowでの入力に対応
- ✓ Apache Arrowでの出力に対応

まずはここまでで十分

Apache Arrow入出力対応のメリット

- ✓ 世の中にはApache Arrow対応システムがすでにたくさんある
 - ✓ pandas, BigQuery, DuckDB, ClickHouse, ...
 - ✓ これらのシステムと低コストで連携できる
- ✓ これからも増えるはず
 - ✓ PostgreSQLにもパッチを送っている
3年くらい前からやっている

入力

```
require "arrow-format"

File.open("input.arrow", "rb") do |input|
  reader = ArrowFormat::FileReader.new(input)
  reader.each do |record_batch|
    record_batch.each_record do |record|
      p record.column1
    end
  end
end
```

入力：Railsアプリ

```
def import
  file = params[:file]
  case File.extname(file.original_filename)
  # CSVをインポートするみたいに
  when ".csv"; import_csv(file)
  # Apache Arrowデータをインポートして欲しい！
  when ".arrow"; import_arrow(file)
  when ".arrows"; import_arrows(file)
  end
end
```

出力

```
require "arrow-format"

File.open("output.arrow", "wb") do |output|
  writer = ArrowFormat::FileWriter.new(output)
  values = {
    integer: [1, 2, 3],
    string: ["Hello", nil, "World"],
  }
  record_batch = ArrowFormat::RecordBatch.new(values)
  writer.start(record_batch.schema)
  writer.write_record_batch(record_batch)
  writer.finish
end
```

出力：Railsアプリ

こんな感じで書けるといい？

```
def index
  items = Item.all
  respond_to do |format|
    format.arrow {render body: items.to_arrow.format(:arrow)}
    format.arrows do
      send_stream do |stream|
        items.to_arrow.format(:arrows, output: stream)
      end
    end
  end
end
```

これが必要：IANAに登録済み。私が申請した！！

```
Mime::Type.register("application/vnd.apache.arrow.file", :arrow)
Mime::Type.register("application/vnd.apache.arrow.stream", :arrows)
```

次の一步のための改良案

- ✓ CSVとかも統一して読み書きできるAPI
- ✓ pure Ruby Apache Parquet実装
 - ✓ Arrowはインメモリー用でParquetは保存用
 - ✓ pure Ruby Apache Thrift実装が必要！
C++実装のバインディングはある
- ✓ Cデータインターフェイス対応
- ✓ pure Rubyデータ処理機能の実装
今はデータの読み書きだけ

Cデータインターフェイス

- ✓ 同一プロセス内でのArrowデータ交換API
<https://arrow.apache.org/docs/format/CDataInterface.html>
- ✓ CのABIとして定義
- ✓ 別のArrow処理系にゼロコピーでデータを渡せる
Go実装↔Rust実装とかができる
- ✓ データのアドレスとメタデータを交換しているだけ
- ✓ 活用例：DuckDBとやりとりできる

pure Rubyでの実装に必要なもの

- ✓ `I0::Buffer`の実データのアドレス
 - ✓ `MemoryView`に対応させて取得しよう
 - ✓ でも、`MemoryView`はRubyじゃなくてCのAPI
 - ✓ `Fiddle`に`MemoryView`のバインディングがある
 - ✓ `Fiddle`を使ってもpure Rubyって言うていい…？
- ✓ 実データのアドレスを`I0::Buffer`で参照
 - ✓ これも`MemoryView`でいけるはず

pure Rubyデータ処理機能

- ✓ Rubyで書いているのに速い！を目指したい
- ✓ JITでSIMDすればいけるのでは。。。？
- ✓ SIMDに必要：
 - ✓ 同一サイズの連続して配置されたデータ
Apache ArrowデータはSIMDで使える
 - ✓ ↑をどうやってJITで検出するの？
 - ✓ Rubyのcoreにはそんなオブジェクトはない

MemoryViewでは？

- ✓ MemoryView :
 - ✓ 同一サイズの要素の多次元配列をゼロコピーで共有するための仕組み
- ✓ `int32_t` とかの型情報も持っている
- ✓ CのAPIなのでRubyレベルには出てこない
 - ✓ JITで検出できない
 - ✓ Fiddle提供のバインディングを検出はアリ。。。？

まとめ

- ✓ みんなも開発助成金制度を活用してね
- ✓ Apache Arrowの入出力に対応させてね
- ✓ 一緒にApache Arrowを使いやすくしない？
- ✓ JITでSIMDっていけると思う？