Apache Arrow, Red Arrow, Red Data Tools

須藤功平

株式会社クリアコード

地球流体データ解析・数値計算ワークショップ

2023-03-29

須藤功平

- ✓ kou
- ✓ 2004年からRubyコミッター
- ✓ 2017年から
  Apache Arrowのコミッター
- ✓ 2017年にRed Data Tools開始
- ✓ 2022年のApache ArrowのPMCチェアー

今日の話

- ✓ あなたたちもRubyでデータ処理できるようにする側にまわるんだよ！！！
- ✓ これからのデータ処理界隈のことを知るにはApache Arrowのことを知っておくといいから基本を説明するね！

Rubyでデータ処理、2023

- ✓ 相変わらずPythonとかRほどはできない
- ✓ でも、できることは増えている

できること

- ✓ 多次元数値配列の演算：Numo::NArray
- ✓ データフレームでの前処理：RedAmber他
- ✓ 可視化：Charty他
- ✓ データセットへのアクセス：Red Datasets
- ✓ 著名データフォーマットの読み書き
- ✓ ...

なぜできることが増えているのか

手を動かしている
人たちが
いるから！

手を動かしている人たち

- ✓ Red Data Tools
- ✓ Ruby Numo (NUmerical MOdule)
- ✓ SciRuby
- ✓ BioRuby
- ✓ ankane
- ✓ ...

あなたがするべきこと

私たちと一緒に
手を動かす！

Red Data Toolsポリシー

1. Rubyコミュニティーを超えて協力する
2. 非難することよりも手を動かすことが大事
3. 一回だけの活発な活動よりも小さくてもいいので
   継続的に活動することが大事
4. 現時点での知識不足は問題ではない
5. 部外者からの非難は気にしない
6. 楽しくやろう！

https://red-data-tools.github.io/ja/

活動例

Apache Arrowの開発

Apache Arrow

インメモリー
データ処理の
共通基盤

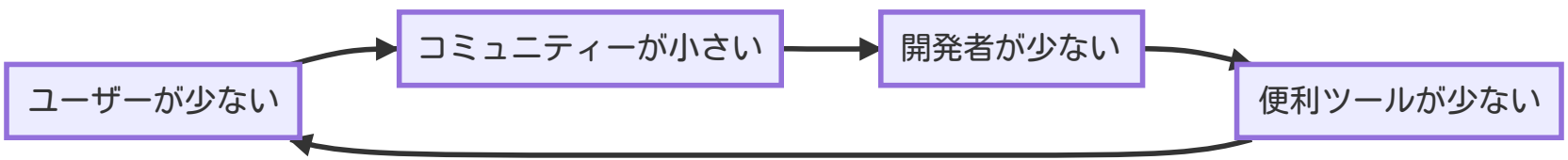
共通基盤

各データ処理プロダクト共通で必要なやつはみんなで協力して作って共有しようぜ！

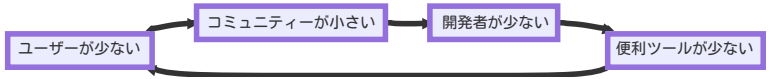
共通で必要なやつ？共有？

- たくさんある！
  - 今どきはPython用に作られていたようなやつら
  - 理由：すでにPythonにエコシステムがあるから
- なんで共有したいの？
  - Python以外でも使いたい！（エコシステムがあるのに！？）
  - RとかJavaとか！最近だとRust！
    Rubyも仲間に入れて！！！

Rubyとデータ処理
負のループ

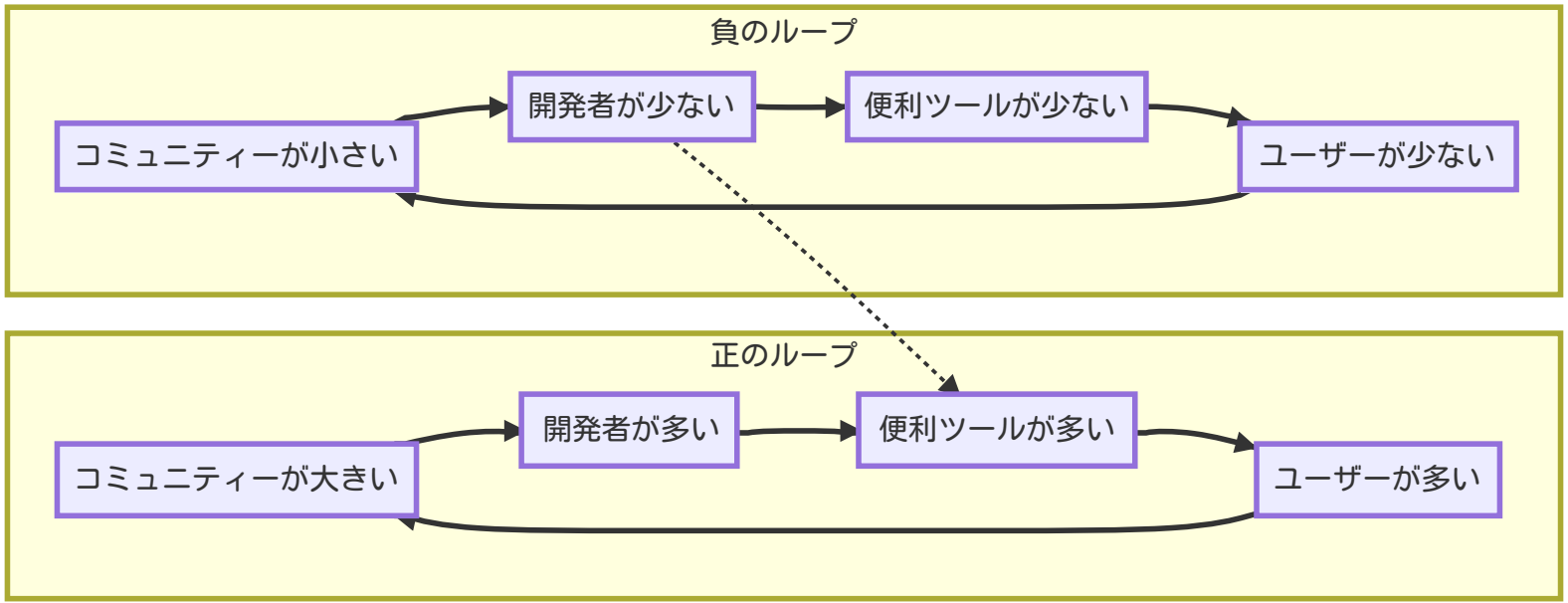


この負のループを打ち破るには？



- ✓ ユーザーが少ない：便利ツールを増やす？
- ✓ コミュニティーが小さい：ユーザー増？
- ✓ 開発者少：コミュニティーを大きくする？
- ✓ 便利ツールが少ない：開発者を増やす？

少ない開発者で便利ツールを増やす



そこでApache Arrowですよ！！！

- ✓ 共通基盤を共有！
- ✓ Rubyも乗るしかない！
  - ✓ 他の言語の人たちと共同開発できる
  - ✓ Ruby開発者が少なくても便利ツールを増やせる

ということでやっている

- ✓ 2017年からApache Arrowのコミッター
- ✓ 2017年にRed Data Tools開始
  - ✓ 1. Rubyコミュニティーを超えて協力する
- ✓ 2022年のApache ArrowのPMCチェアー

成果例

- ✓ Red Arrow
  - ✓ Apache ArrowのRubyバインディング
- ✓ RedAmber
  - ✓ Red Arrowベースのデータフレーム
- ✓ Red Parquet
  - ✓ Apache Parquetの読み書き

Apache Arrowが提供するもの

- ✓ 読み書きがすごく速いデータフォーマット
- ✓ ↑を活かしたRPCフレームワーク
- ✓ ↑を活かしたDB連携機能
- ✓ 各種データフォーマットとの相互変換
- ✓ 計算がすごく速いメモリーレイアウト
- ✓ ↑を活かした速い計算
- ✓ ...

すごく速いデータフォーマット

ClearCode

ClearCode

Apache Arrowフォーマットは なぜ速いのか

須藤功平
株式会社クリアコード

db tech showcase ONLINE 2020
2020-12-08

https://slide.rabbit-shocker.org/authors/kou/db-tech-showcase-online-2020/

Apache Arrowフォーマット

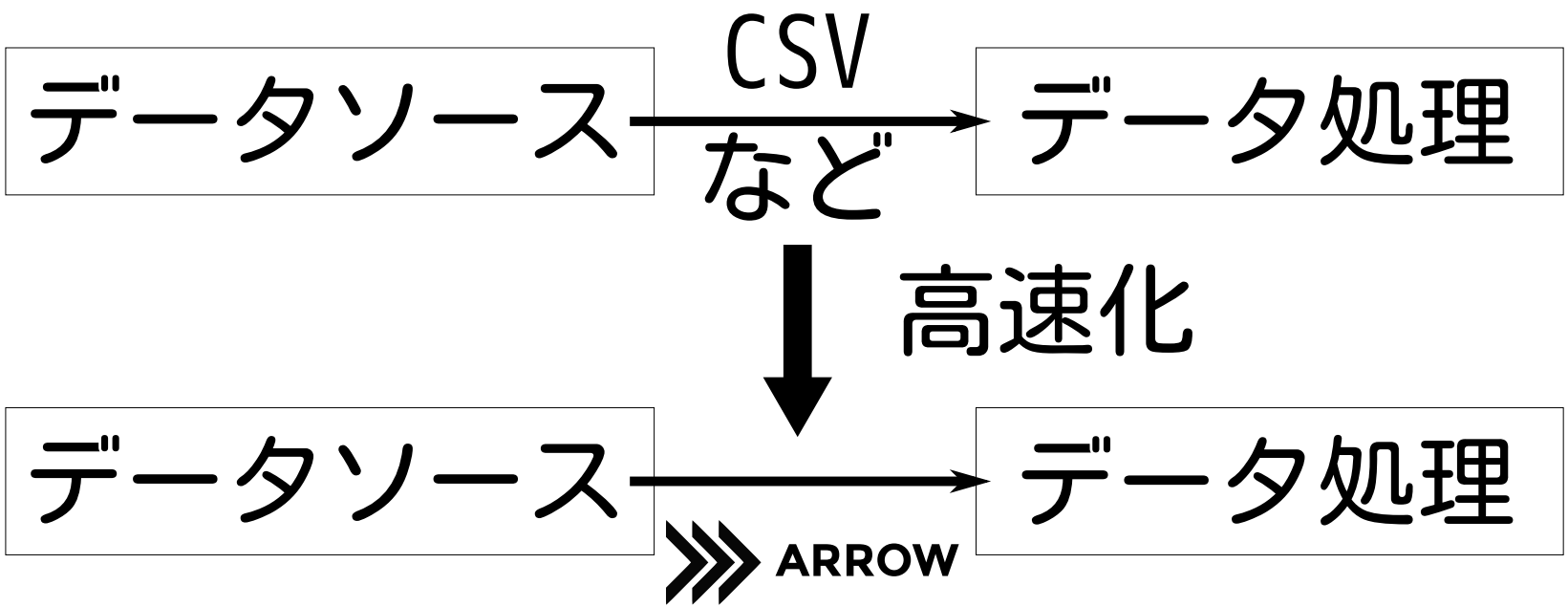
- データフォーマット
  - 通信用とインメモリー用の両方
- 表形式のデータ用
  - ＝データフレーム形式のデータ用
- 速い！

速い！

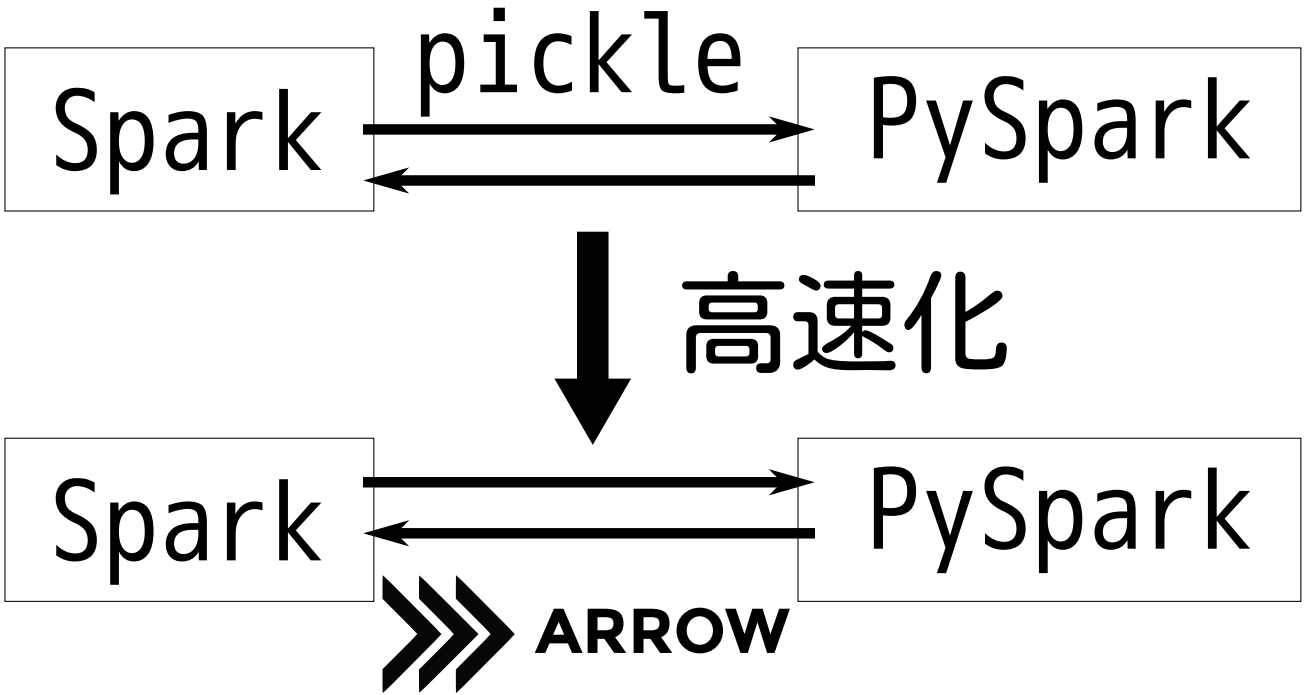
- ✓ データ**交換**が速い！
- ✓ データ**処理**が速い！

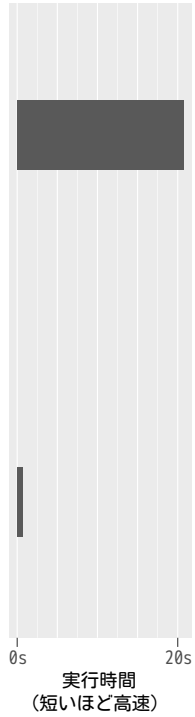
データ交換が速い！

Apache Arrowフォーマットにすると高速化！



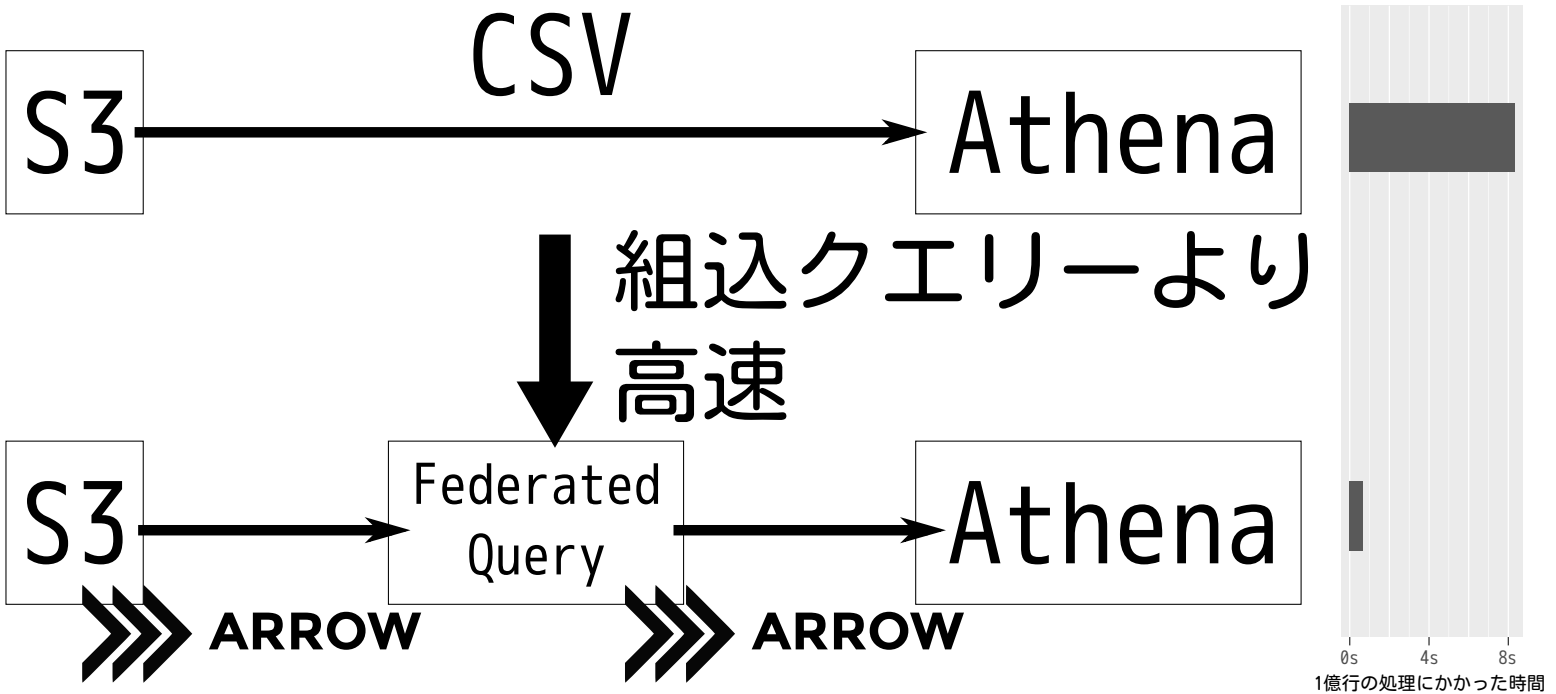
利用事例：Apache Spark





Data at https://arrow.apache.org/blog/2017/07/26/spark-arrow/

利用事例：Amazon Athena



Data at https://github.com/awslabs/aws-athena-query-federation/tree/master/athena-federation-sdk#performance

どうして速いの？

- ✓ シリアライズコストが低い
  - ✓ すぐに送れるようになる
- ✓ デシリアライズコストが低い
  - ✓ 受け取ったデータをすぐに使えるようになる

シリアライズ処理

1. メタデータを用意
2. メタデータ+元データそのものを送信
   - ✓ 元データを加工しないから速い！
   - ✓ なにもしないのが最速！

元データを加工する例：JSON

```
0x01 0x02（8bit数値の配列）
    ↓
"[1,2]"（JSON）
0x01→0x49（数値→ASCIIの文字：'1'）
0x02→0x50（数値→ASCIIの文字：'2'）
```

元データそのものを使うと…

- ✓ 変換処理にCPUを使わなくてよい
  - ✓ 速い
- ✓ 変換後のデータ用のメモリー確保ゼロ
  - ✓ 大きなメモリー確保はコストが高い
  - ✓ 一定の作業領域を使い回すとかしなくてよい
  - ✓ 速い

デシリアライズ処理

1. メタデータをパース
2. メタデータを基に元データを取り出す
   - ✓ 元データをそのまま使えるから速い！
   - ✓ なにもしないのが最速！

元データを元に戻す例：JSON

```
"[1,2]" (JSON)
    ↓
0x01 0x02 (8bit数値の配列)
0x49→0x01 (ASCIIの文字：'1'→数値)
0x50→0x02 (ASCIIの文字：'2'→数値)
```

元データを取り出せると…

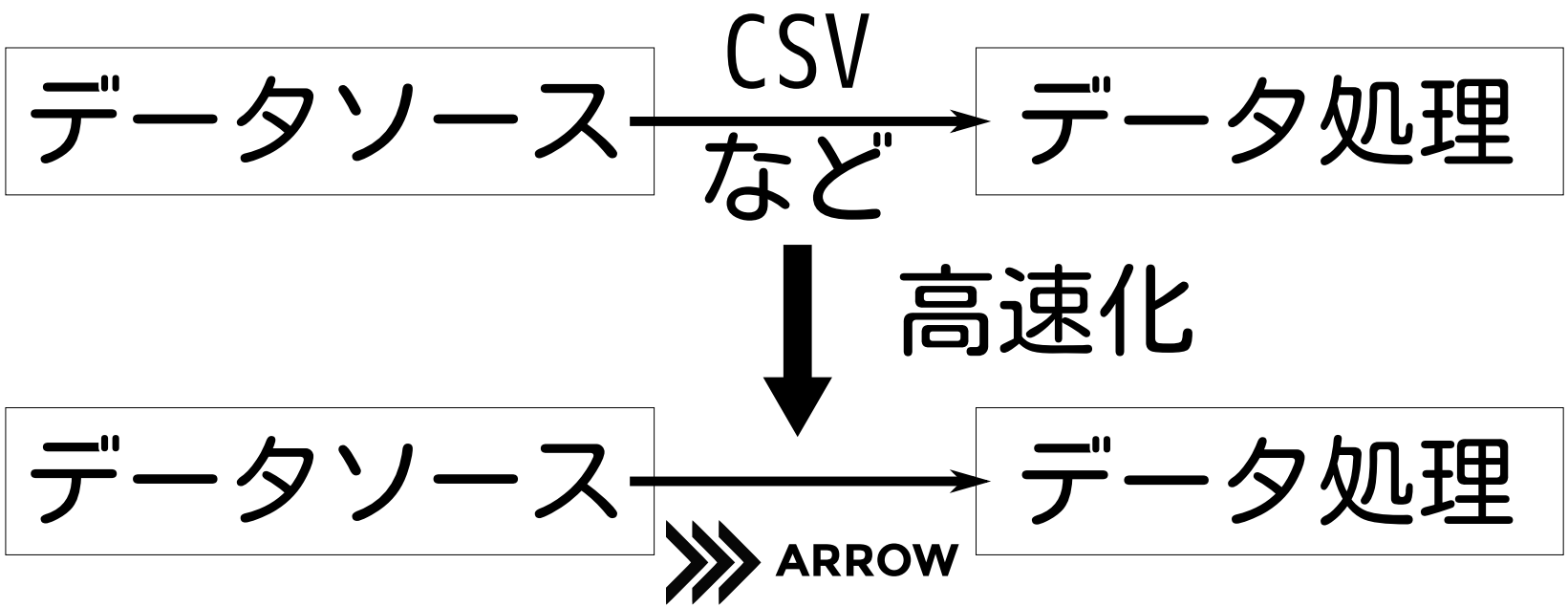
- ✓ 変換処理にCPUを使わなくてよい
  - ✓ 速い
- ✓ 変換後のデータ用のメモリー確保ゼロ
  - ✓ すでにあるデータをそのまま使うのでゼロコピー
  - ✓ 速い
- ✓ メモリーマップで直接データを使える
  - ✓ ディスク上のメモリー以上のデータを扱える

{,デ}シリアライズコスト

- ✓ Apache Arrowフォーマット
  - ✓ ほぼメタデータのパースコストだけ
- ✓ それ以外の多くのフォーマット
  - ✓ データ変換処理（CPU）
  - ✓ 作業用メモリー確保処理（メモリー）

データ交換が速い！

Apache Arrowフォーマットにすると高速化！



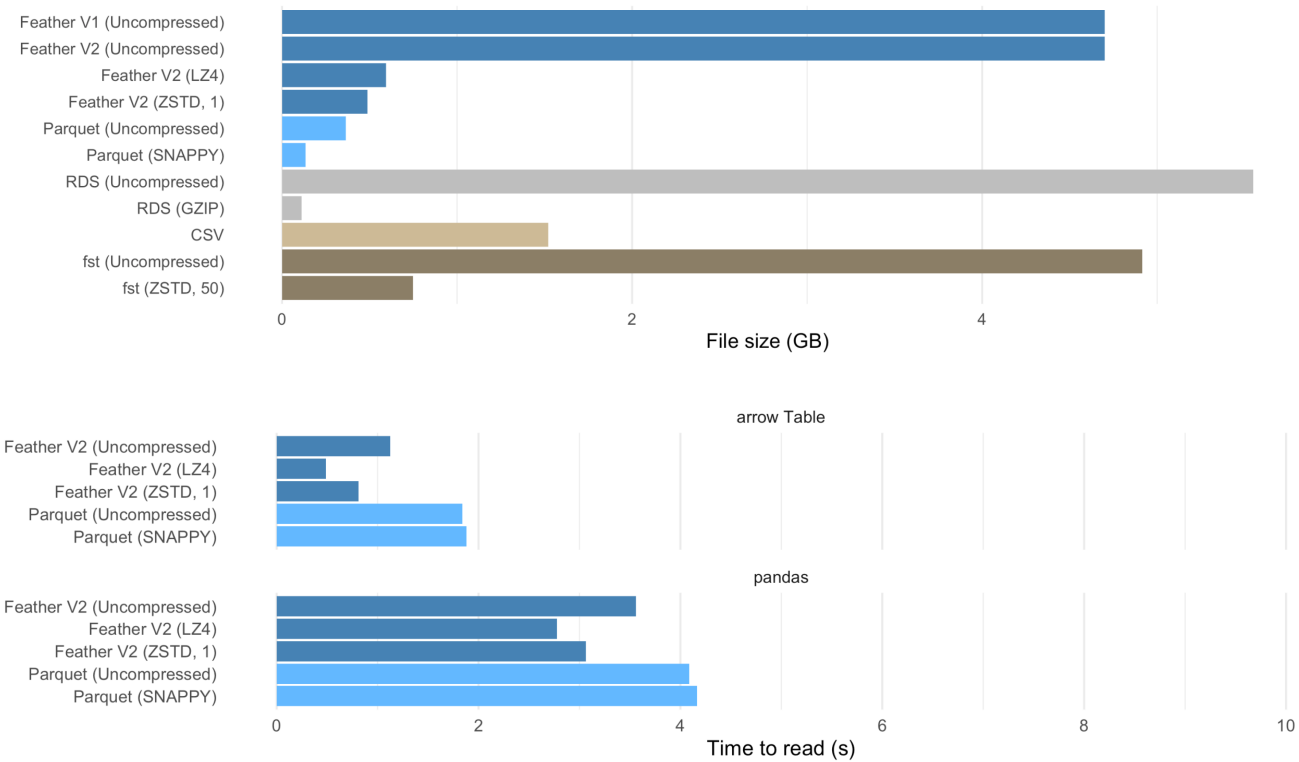
データサイズは？

- ✓ CPU・メモリーにやさしくて速いんだね！
- ✓ じゃあ、データサイズはどうなの？
  - ✓ 大きいとネットワーク・IOがボトルネック
  - ✓ 本来のデータ処理を最大限リソースを使いたい
  - ✓ データ交換をボトルネックにしたくない

データサイズ

- ✓ 別に小さくない
  - ✓ 元データそのままなのでデータ量に応じて増加
- ✓ Zstandard・LZ4での圧縮をサポート
  - ✓ ゼロコピーではなくなるがサイズは数分の1
  - ✓ 圧縮・展開が必要→元データそのものを使えない
  - ✓ CPU・メモリー負荷は上がるが
ネットワーク・IO負荷は下がる
  - ✓ ネットワーク・IOがボトルネックになるなら効く

圧縮時のデータサイズと読み込み速度



https://ursalabs.org/blog/2020-feather-v2/

データ交換が速い！のまとめ

- ✓ {,デ}シリアライズが速い
  - ✓ 元データをそのまま使うので処理が少ない
  - ✓ CPU・メモリーにやさしい
- ✓ 圧縮もサポート
  - ✓ ネットワーク・IOがボトルネックならこれ
  - ✓ CPU・メモリー負荷は上がるが
    データ交換のボトルネックを解消できるかも

交換したデータの扱い

- ✓ データ分析はデータ交換だけじゃない
  - ✓ データ交換だけ速くしても基盤とは言えない
- ✓ データ処理も速くしないと！
  - ✓ データ処理を速くするにはデータ構造が大事

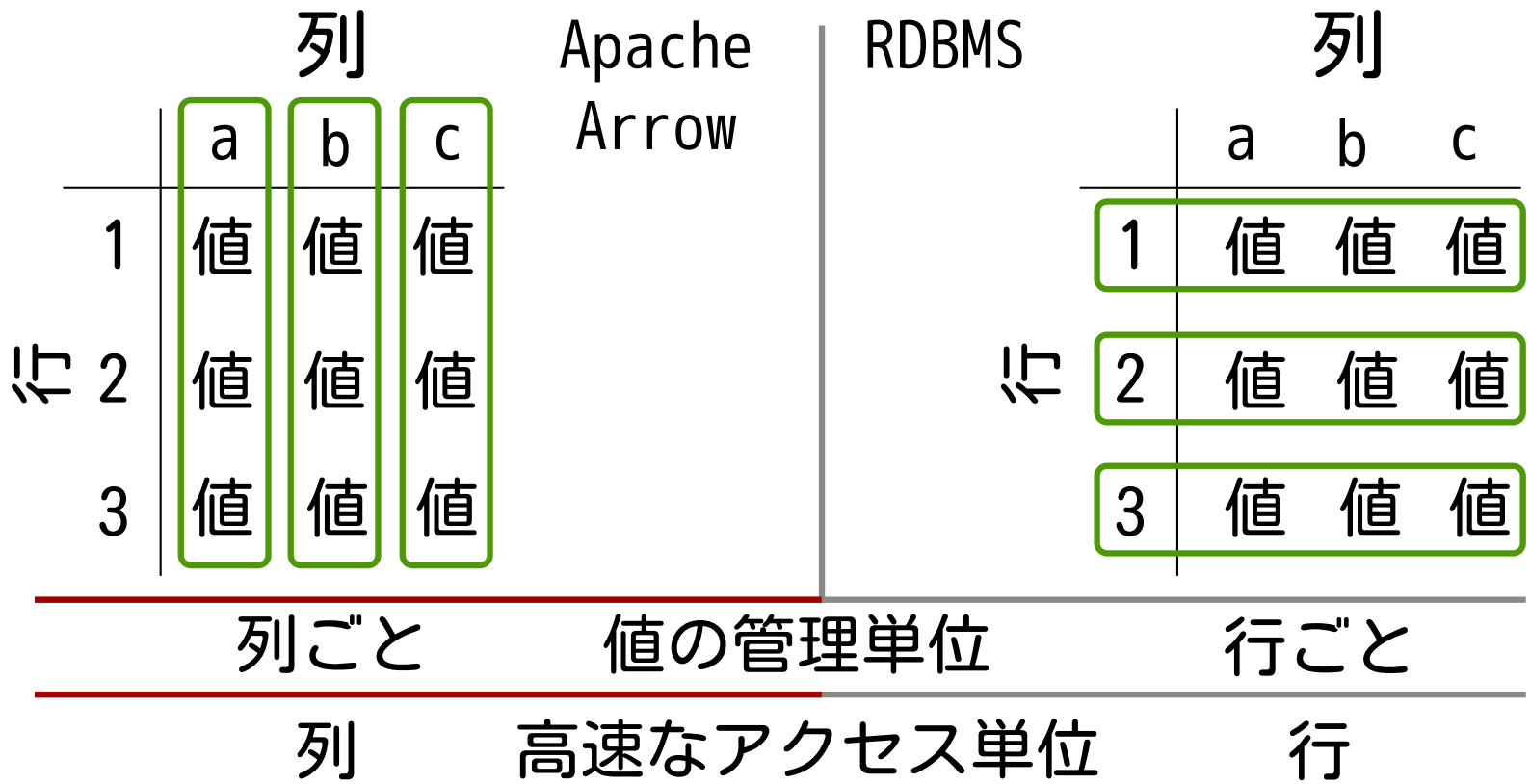
高速処理のためのデータ構造

- 基本方針：
  - 関連するデータを近くに置く
- 効果：
  - CPUキャッシュミスを減らす
  - SIMDを活用できる

データ分析時の関連データ

- ✓ 分析時はカラムごとの処理が多い
  - ✓ 集計・ソート・絞り込み…
- ✓ 同じカラムのデータを近くに置く
  - ✓ カラムナーフォーマット

カラムナーフォーマット



各カラムでのデータの配置

- ✓ 関連するデータを近くに置く
- ✓ 定数時間でアクセスできるように置く
- ✓ SIMDできるように置く
  - ✓ アラインする
    アライン：データの境界を64の倍数とかに揃える
  - ✓ 条件分岐をなくす

真偽値・数値のデータの配置

固定長データなので連続して配置
32ビット整数：[1, 2, 3]
0x01 0x00 0x00 0x00 0x02 0x00 0x00 0x00 0x03 ...

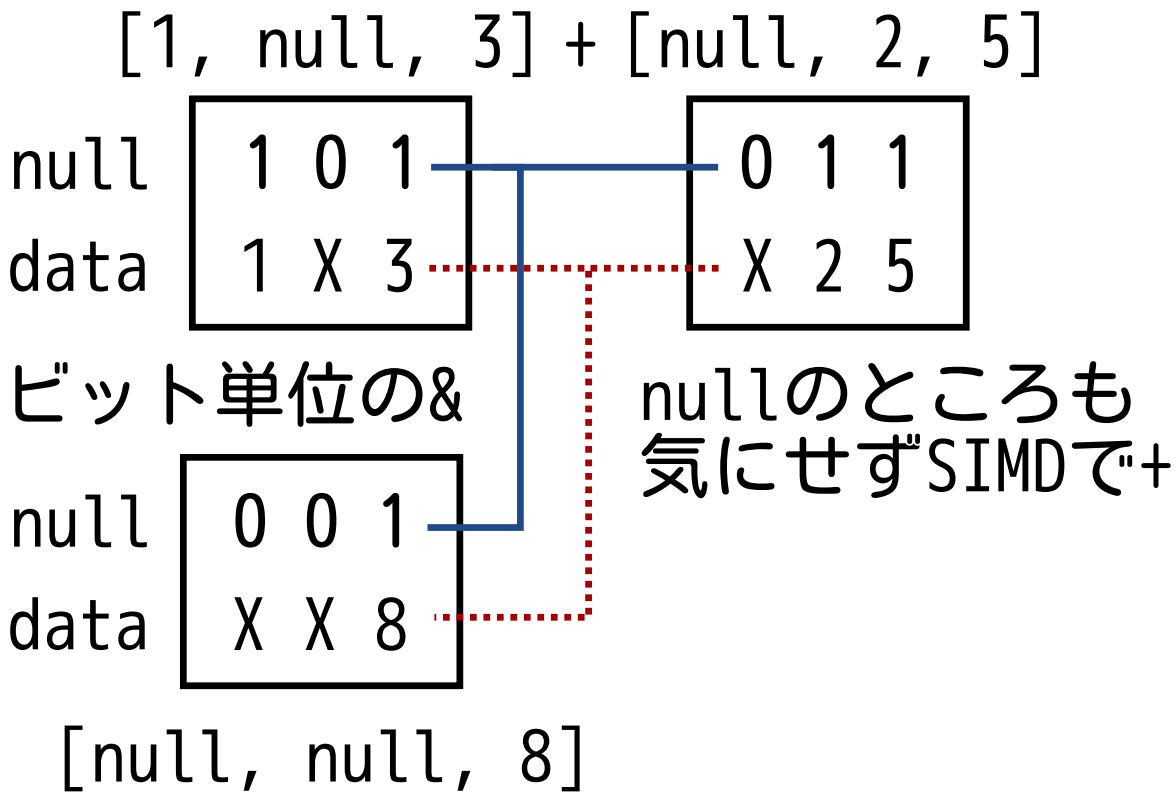
文字列・バイト列：データの配置

実データバイト列+長さ配列に配置
UTF-8文字列：["Hello", "", "!"]
実データバイト列："Hello!"
長さ配列：[0, 5, 5, 6]
i番目の長さ：長さ配列[i+1] - 長さ配列[i]
i番目のデータ：
　実データバイト列[長さ配列[i]..長さ配列[i+1]]
注：長さ→データ→長さ→データ→…で置くと
　　定数時間でi番目にアクセスできない

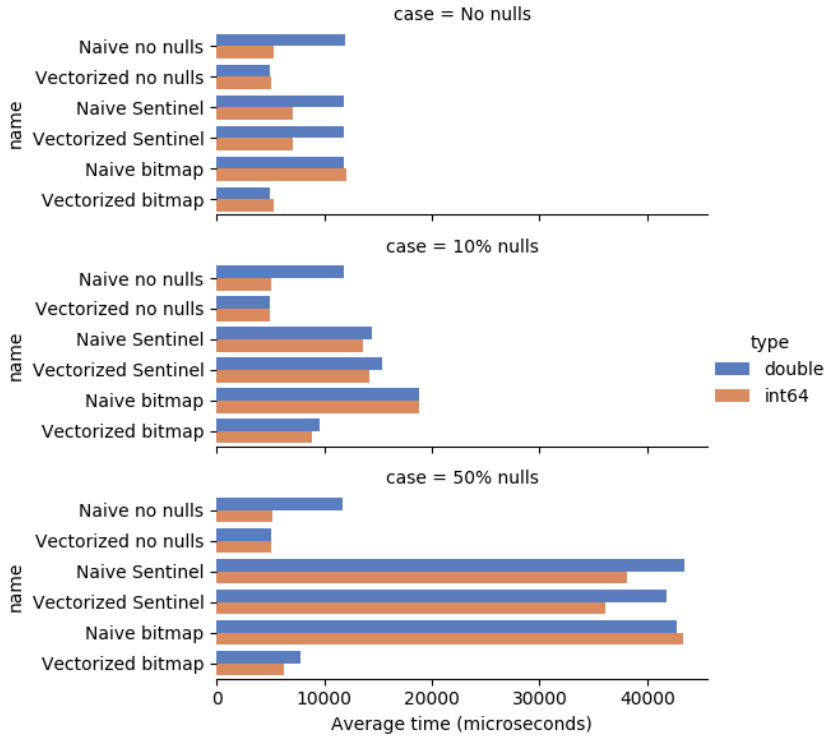
nullと条件分岐

- ✓ null対応のアプローチ：
  - ✓ null値：Julia: missing, R: NA
  - ✓ 別途ビットマップを用意：Apache Arrow
- ✓ ビットマップを使うと条件分岐をなくせる

nullと条件分岐とSIMD



nullと条件分岐とSIMD



https://wesmckinney.com/blog/bitmaps-vs-sentinel-values/

高速なデータ処理のまとめ

- ✓ 高速なデータ処理にはデータ構造が重要
  - ✓ データ分析にはカラムナーフォマットが適切
- ✓ 定数時間でアクセス可能なデータの配置
- ✓ SIMDにやさしいデータの持ち方
  - ✓ アライン・null用のビットマップ

まとめ：Apache Arrowフォーマット

- ✓ Apache Arrowフォーマット
  - ✓ 通信用・インメモリー用のデータフォーマット
  - ✓ 表形式のデータ用
- ✓ Apache Arrowフォーマットは速い
  - ✓ データ**交換**が速い
  - ✓ データ**処理**が速い
  - ✓ データ交換してすぐに高速処理できるフォーマット

まとめ：なぜデータ交換が速いのか

- ✓ 元データをそのままやりとりできるから
  - ✓ {,デ}シリアライズコストが低い
  - ✓ CPU・メモリーにやさしい
- ✓ ネットワーク・IOの負荷を下げたい
  - ✓ Zstandard・LZ4による圧縮
  - ✓ CPU・メモリー負荷は上がるが
ネットワーク・IO負荷は下がる

まとめ：なぜデータ処理が速いのか

- ✓ 最適化されたデータ構造
  - ✓ カラムナーフォーマット
  - ✓ SIMDを使えるデータの持ち方
- ✓ 最適化された実装
  - ✓ 今回は紹介していない！！！
  - ✓ pandas 2.0.0はこれを使って高速になるよ！

Red Arrow

- Apache ArrowのRubyバインディング
  - Apache Arrowの機能をRubyらしいAPIで
  - red-parquet/red-adbcなど別gemに分離した機能も
- 既存gemのApache Arrow対応は別gemで
  - red-arrow-activerecord: Active Recordのクエリー結果をRed Arrowオブジェクトに変換
  - red-arrow-duckdb: DuckDBの入出力をRed Arrowオブジェクトで

あなたとRed Arrow

- よく使うだろう機能：
  - Arrow::Table.load(path)：いい感じに読み込む
  - Arrow::Table#save(path): いい感じに書き出す
- 開発に参加！は難しそう
  - 使用技術：C++/GObject Introspection/Ruby（少し）

Red Data Tools

Rubyでデータ処理
できるように
するための
プロジェクト

Red Data Toolsプロダクツ

- ✓ Red Arrow
- ✓ Charty：可視化
- ✓ Red Datasets：データセット
- ✓ RedAmber：データフレーム
- ✓ ...

あなたとRed Data Tools

- ✓ 使う
- ✓ 開発に参加する
  - ✓ 4. 現時点での知識不足は問題ではない
  - ✓ 6. 楽しくやろう！
- ✓ チャットにおいで！
  - ✓ https://gitter.im/red-data-tools/ja

あなたとRed Data Toolsプロダクツ

- ✓ Red Arrow：難しそう
- ✓ Charty：いけるかも？
- ✓ Red Datasets：よさそう
- ✓ RedAmber：よさそう
- ✓ ...（チャットでよさそうなプロダクトを一緒に考えるよ！）...