Apache Arrow 2019 #ArrowTokyo

須藤功平
株式会社クリアコード

Apache Arrow東京ミートアップ2019
2019-12-11

Apache Arrowと私

- ✓ 2016-12-21に最初のコミット
- ✓ 2017-05-10にコミッター
- ✓ 2017-09-15にPMCメンバー
- ✓ 2018-12-06現在コミット数3位（224人中）
- ✓ 2019-12-09現在コミット数2位（348人中）

Apache Arrow 1.0.0がでるぞ！

- ✓ たぶん2020年1月か2月あたり
- ✓ 使い始めるなら今！
  - ✓ 普通のユーザー：リリースされたら試そう
  - ✓ 先進的なみなさん：リリースされたら本格使用！

今日の目的

Apache Arrow
ユーザーを
増やす

今日のチャレンジ

こんな人たちもフォローしてユーザーに！

- ✓ 使ってみたい！
- ✓ でも、自分たちだけじゃ不安…

対策：みんなで力を合わせる！

こんな布陣ならいけるかも！なら相談して！

https://www.clear-code.com/contact/

- ✓ クリアコード：技術提供（すごく詳しいよ！）
- ✓ A社：データ提供（うちのデータでApache Arrowは効くかな？）
- ✓ B社：インフラ提供（どんな構成が適切かな？）
- ✓ C社：お金・人的リソース提供（知見を貯めたい）
- ✓ ...

Apache Arrow

各種言語で使えるインメモリーデータ処理プラットフォーム

実現すること

データ処理の効率化
（大量データが対象）

効率化のポイント

- ✓ 速度
  - ✓ 速いほど効率的
- ✓ 実装コスト
  - ✓ 低いほど効率的

速度向上方法

- ✓ 遅い部分を速く
- ✓ 高速化できる部分を最適化

遅い部分

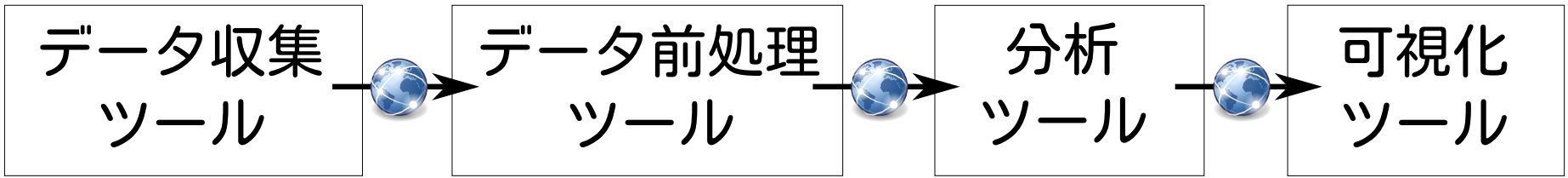
データ交換

データ交換

- ✓ データ処理ツール間で必要
- ✓ データ処理システム
  - ✓ 複数ツールを組み合わせて実現
  - ✓ データ処理システムではデータ交換が必須

データ処理システム例

- データ交換



データ交換処理

1. シリアライズ
   データをバイト列へ変換
2. 転送
   バイト列を別ツールに渡す
3. デシリアライズ
   バイト列からデータを復元

データ交換処理：必要なリソース

1. シリアライズ：CPU
2. 転送：I/O（ネットワーク・ストレージ・メモリー）
3. デシリアライズ：CPU

データ交換の高速化

- ✓ データ量が増加すると
  シリアライズ・デシリアライズ速度が劣化
- ✓ 速度劣化を抑えられれば高速化可能

Apache Arrowのアプローチ

なにもしなければ最速！

- ✓ データフォーマットを定義
  - ✓ ほぼパースいらずなので速い！
  - ✓ シリアライズ・デシリアライズが高速化！
- ✓ このフォーマットを普及
  - ✓ 各種言語で読み書き処理を実装
  - ✓ みんなが使えばフォーマット変換いらずで速い！

Apache Sparkでの高速化事例

- ✓ Spark⇔PySpark間でデータ交換
  - ✓ Apache Arrowを使うことで数十倍レベルの高速化
- ✓ Spark⇔R間でデータ交換
  - ✓ sparklyrでは10倍以上の高速化
  - ✓ Spark 3.0からはSparkRでも使える

Spark関連は山室さんが紹介！

Apache Arrowのさらなるアプローチ

1. シリアライズ
   データをバイト列へ変換
2. 転送←ここも高速化
   バイト列を別ツールに渡す
3. デシリアライズ
   バイト列からデータを復元

同一ホスト時のデータ転送の高速化

- メモリーマップ機能
  - ファイルの内容をメモリー上のデータのようにアクセスできる機能
  - readせずにデータを使える（データコピー不要）
- パース不要+メモリーマップ
  - デシリアライズ時にメモリー確保不要
  - 「転送」コスト削減

複数ホスト時のデータ転送の高速化

- RPCフレームワークを提供
  - Apache Arrow Flight
  - gRPCベース
- 詳細：Apache Arrow Flightの紹介

効率的なデータ交換処理のまとめ

- ✓ 高速なデータフォーマット
- ✓ 効率的なデータ交換処理
  - ✓ **同一ホスト内**での高速なデータ交換
  - ✓ **異なるホスト間**での高速なデータフレーム交換

速度向上方法

- ✓ 遅い部分を速く
  - ✓ **データ交換**を速く
- ✓ 高速化できる部分を最適化

高速化できる部分

大量データの計算

大量データの計算の高速化

- ✓ 各データの計算を高速化
- ✓ まとまったデータの計算を高速化

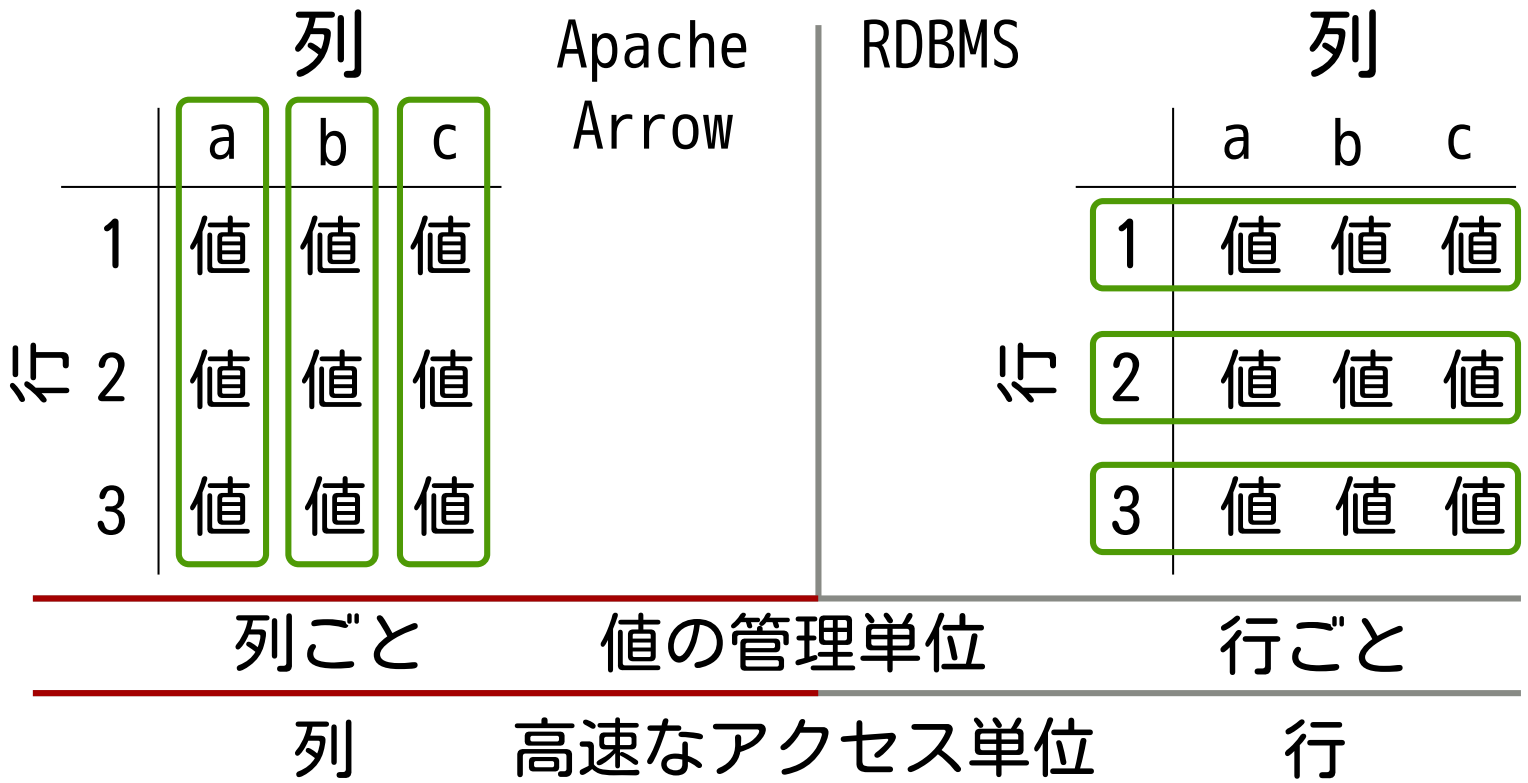
各データの計算の高速化

- ✓ データを局所化
  - ✓ CPUのキャッシュメモリーを活用できる
- ✓ 局所化に必要な知識
  - ✓ データの使われ方
  - ✓ 局所化：一緒に使うデータを近くに置く

想定ユースケース

- OLAP (OnLine Analytical Processing)
  - データから探索的に知見を探し出すような処理
- 列単位の処理が多い
  - 集計処理・グループ化・ソート…

OLAP向きのデータの持ち方



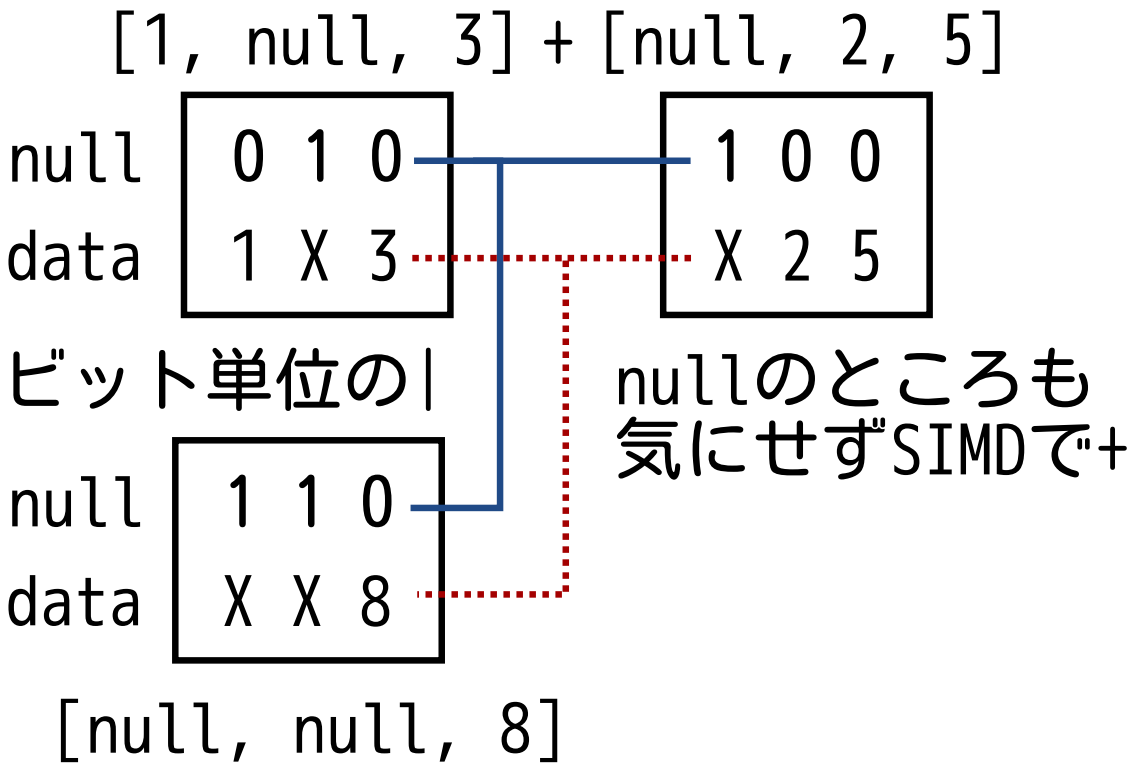
まとまったデータの計算を高速化

- ✓ SIMDを活用
  Single Instruction Multiple Data
- ✓ スレッドを活用
- ✓ ストリームで処理

SIMDを活用

- ✓ CPU：データをまとめてアライン
  アライン：データの境界を64の倍数とかに揃える
- ✓ GPUの活用
- ✓ 条件分岐をなくす
  - ✓ null/NA用の値は用意せずビットマップで表現
    Is it time to stop using sentinel values for null / NA values?

条件分岐とnull


[1, null, 3] + [null, 2, 5]
null
data
0 1 0
1 X 3
1 0 0
X 2 5
ビット単位の|
nullのところも
気にせずSIMDで+
null
data
1 1 0
X X 8
[null, null, 8]

スレッド活用時のポイント

- ✓ 競合リソースを作らない
  - ✓ リソースロックのオーバーヘッドで遅くなりがち
- ✓ アプローチ
  - ✓ リソースを参照するだけ
  - ✓ 各スレッドにコピー

Apache Arrowとスレッド

- ✓ データはリードオンリー
  - ✓ スレッド間でオーバーヘッドなしで共有可能
- ✓ データコピーは極力避けたい
  - ✓ データ交換時もスレッド活用時も

ストリームで処理

- CPUを遊ばせない
  - データ読み込み中にすでに読んだデータを処理
- C++実装：詳細は村田さんが紹介！
  - Apache Arrow C++ Datasets：データ読み込み
  - Apache Arrow C++ Query Engine：データ処理
  - Apache Arrow C++ Data Frame：高レベルAPI
  - Gandiva：高速な式コンパイラー

高速化のまとめ

- ✓ 速度
  - ✓ 遅い処理（**データ交換処理**）を高速化
  - ✓ 速くできる処理（**大量データの計算**）を最適化
- ✓ 実装コスト
  - ✓ 低いほど効率的

実装コストを下げる

- ✓ 共通で使いそうな機能をライブラリー化
  - ✓ メリットを受ける人たちみんなで協力して開発
  - ✓ 最適化もがんばる
- ✓ Apache Arrowの実装コストは下がらない
  - ✓ Apache Arrowを使うツールの実装コストが下がる
    実装コストが下がる：ツール開発者のメリット

共通で使いそうな機能

- ✓ 高速なデータフォーマット（説明済み）
- ✓ 効率的なデータ交換処理（説明済み）
- ✓ 高速なデータ処理ロジック（説明済み）
- ✓ フォーマット変換機能

実装コストのまとめ

- ✓ 速度
  - ✓ 遅い処理（データ交換処理）を高速化
  - ✓ 速くできる処理（大量データの計算）を最適化
- ✓ 実装コスト
  - ✓ 共通で使いそうな機能を**ライブラリー化**
  - ✓ **みんなで協力**して開発

フォーマット変換機能

- Apache Arrowフォーマット
  - インメモリー用のフォーマット
  - データ交換・処理向きで永続化向きではない
- 永続化されたデータを使う場合
  - 永続化に適したフォーマットで保存
  - 読み込み時にApache Arrowに変換して
    インメモリーでの処理にApache Arrowを使う

対応フォーマット：CSV

- ✓ よく使われているフォーマット
- ✓ 亜種が多くてパースが大変
- ✓ C++実装はすごく速い！
  - ✓ **CSV読み込みの高速化でも使える！**

対応フォーマット：Apache Parquet

- ✓ 永続化用フォーマット
  - ✓ 列単位でデータ保存：Apache Arrowと相性がよい
- ✓ 小さい
  - ✓ 列単位の圧縮をサポート
- ✓ 速い
  - ✓ 必要な部分のみ読み込める（I/Oが減る）

対応フォーマット：Apache ORC

- 永続化用フォーマット
  - 列単位でデータ保存：Apache Arrowと相性がよい
  - Apache Parquetに似ている
- Apache Hive用に開発
  - 今はHadoopやSparkでも使える

対応フォーマット：Feather

- 永続化用フォーマット
  - 列単位でデータ保存：Apache Arrowと相性がよい
  - データフレームを保存
- PythonとR間のデータ交換用
- **今は非推奨！**
  - Apache Arrowを使ってね

豆知識：Feather 2

- 名前を再利用するかも
  - [ARROW-5512] Feather V2
- Feather 2: Apche Arrow IPC File Format
  - Apache Arrow IPC Format：FileまたはStreaming
  - Fileの方をFeather 2と呼んじゃう？

対応フォーマット：JSON

- ✓ よく使われているフォーマット
- ✓ これまではテストのために内部使用
- ✓ 正式機能になった

フォーマット変換機能まとめ

- ✓ 高速なデータフォーマット
- ✓ 効率的なデータ交換処理
- ✓ 高速なデータ処理ロジック
- ✓ フォーマット変換機能
  - ✓ **永続化データ**を処理するために必要

対応言語

- ✓ C, C#, C++, Go, Java, JavaScript, Lua
- ✓ MATLAB, Python, R, Ruby, Rust

非公式実装：

- ✓ Julia (Arrow.jl)

実装方法

- ネイティブ実装
  - C#, C++, Go, Java, JavaScript, Julia, Rust
  - その言語になじむ
- C++バインディング
  - C, Lua, MATLAB, Python, R, Ruby
    R関連は湯谷さんが紹介！
  - ホスト言語が遅めでも速い実装になる

各言語のデータ交換対応状況

- ✓ 基本的な型はすべての言語で対応済み
  - ✓ すでに**データ交換用途に使える**
- ✓ 複雑な型は未対応の言語あり
  List, Structなど
  - ✓ 例：C#のList実装は網屋の橋田さんが改良中

各言語のデータ処理対応状況

- ✓ データの高速な計算は一部言語で対応
  - ✓ C++ベース、Java、Rust（DataFusion）、Go
- ✓ フォーマット変換も各言語の対応は様々

ユースケース（1）

- ✓ PostgreSQLの高速化
  - ✓ 海外さんがPG-Stromの事例を紹介！
- ✓ TensorFlow・BigQuery Storage APIがApache Arrowをサポート
  - ✓ 石崎さんと漆山さんが紹介！

ユースケース（2）

- Fluentdでログデータを Apache Arrow化
  - プラグインあり
- Groongaに高速データロード
  - Groonga：全文検索エンジン
  - Fluentdで集めたログデータを
    Apache Arrowで流し込んで高速検索

今日の目的

Apache Arrow
ユーザーを
増やす

今日のチャレンジ

こんな人たちもフォローしてユーザーに！

- ✓ 使ってみたい！
- ✓ でも、自分たちだけじゃ不安…

対策：みんなで力を合わせる！

こんな布陣ならいけるかも！なら相談して！

https://www.clear-code.com/contact/

- ✓ クリアコード：技術提供（すごく詳しいよ！）
- ✓ A社：データ提供（うちのデータでApache Arrowは効くかな？）
- ✓ B社：インフラ提供（どんな構成が適切かな？）
- ✓ C社：お金・人的リソース提供（知見を貯めたい）
- ✓ ...