

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

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
- ✓ 2020-11-25現在コミット数2位 (508人中)

Apache Arrow

- ✓ データ分析ツールの基盤を提供
- ✓ ツールで必要になるやつ全部入り
- ✓ 各種プログラミング言語をサポート

全部ってなに！？

データフォーマットとかそのデータを高速処理する機能とか他の各種データフォーマットと変換する機能とかローカル・リモートにあるデータを透過的に読み書きする機能とか高速RPCとかなんだけど、全部を説明すると「すごそうだけどよくわからないね！」と言われる！

今日のトピック

Apache Arrow フォーマット

Apache Arrowフォーマット

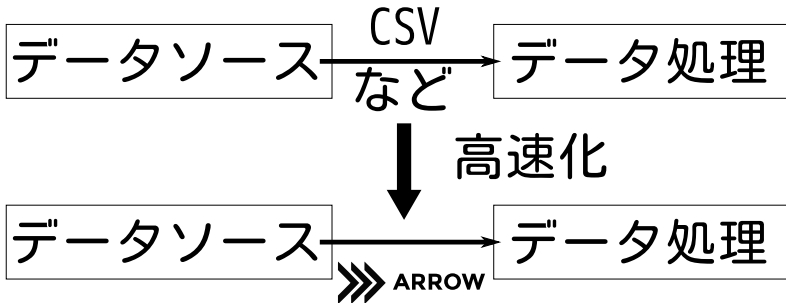
- ✓ データフォーマット
 - ✓ 通信用とインメモリー用の両方
- ✓ 表形式のデータ用
 - ✓ =データフレーム形式のデータ用
- ✓ 速い！

速い！

- ✓ データ**交換**が速い！
- ✓ データ**処理**が速い！

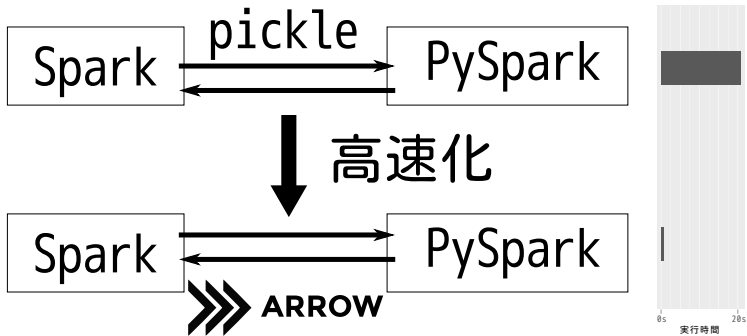
データ交換が速い！

Apache Arrowフォーマットにすると高速化！



Apache Arrow logo is licensed under Apache License 2.0 © 2016-2020 The Apache Software Foundation

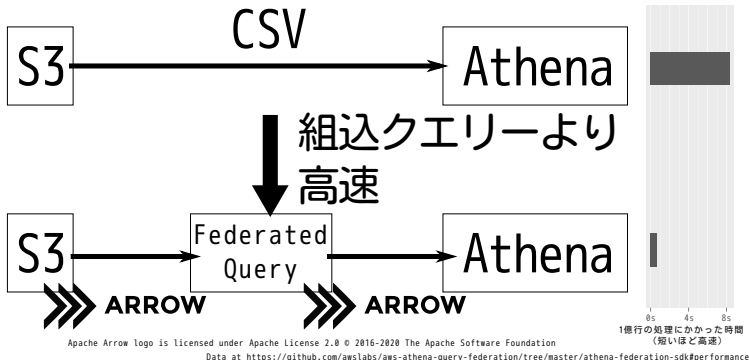
利用事例：Apache Spark



Apache Arrow logo is licensed under Apache License 2.0 © 2016-2020 The Apache Software Foundation

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

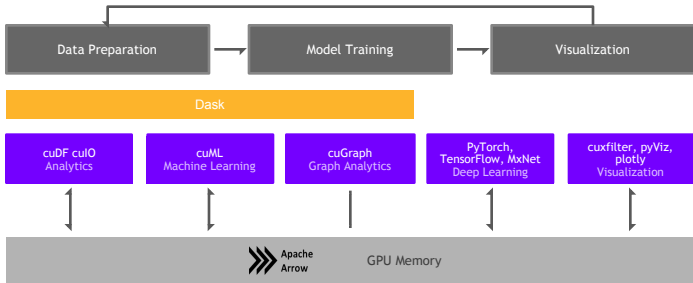
利用事例：Amazon Athena



利用事例：RAPIDS

RAPIDS

End-to-End GPU Accelerated Data Science

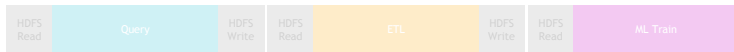


<https://docs.rapids.ai/overview/RAPIDS%20.15%20Release%20Deck.pdf#page=3>

利用事例：RAPIDS

Data Processing Evolution Faster Data Access, Less Data Movement

Hadoop Processing, Reading from Disk



Spark In-Memory Processing



25-100x Improvement
Less Code
Language Flexible
Primarily In-Memory

Traditional GPU Processing



5-10x Improvement
More Code
Language Rigid
Substantially on GPU

RAPIDS



50-100x Improvement
Same Code
Language Flexible
Primarily on GPU

<https://docs.rapids.ai/overview/RAPIDS%20.15%20Release%20Deck.pdf#page=8>

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...

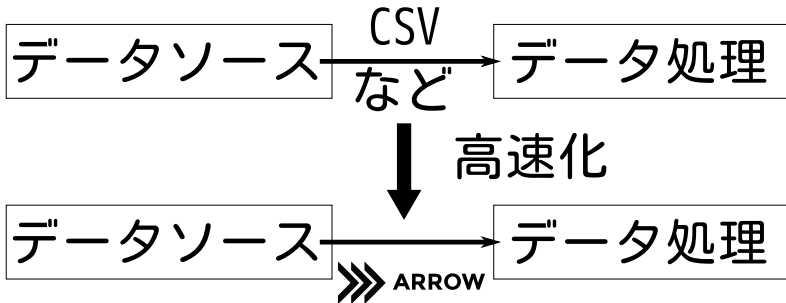
- ✓ 変換処理にCPUを使わなくてよい
 - ✓ 速い
- ✓ 変換後のデータ用のメモリー確保ゼロ
 - ✓ すでにあるデータをそのまま使うのでゼロコピー
 - ✓ 速い
- ✓ メモリーマップで直接データを使える
 - ✓ ディスク上のメモリー以上のデータを扱える

{, 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)

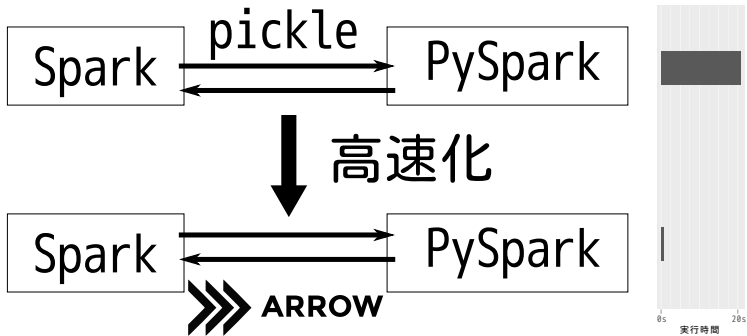
データ交換が速い！

Apache Arrowフォーマットにすると高速化！



Apache Arrow logo is licensed under Apache License 2.0 © 2016-2020 The Apache Software Foundation

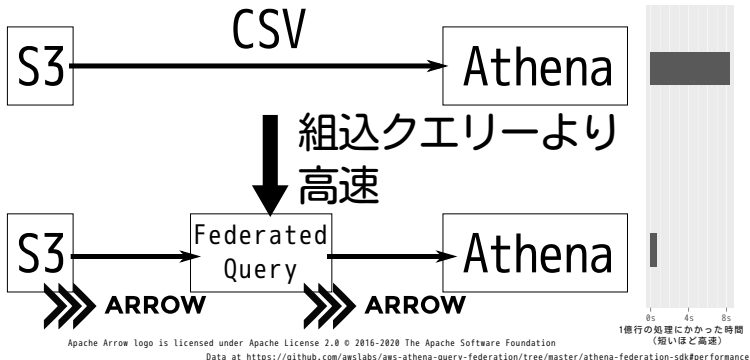
利用事例：Apache Spark



Apache Arrow logo is licensed under Apache License 2.0 © 2016-2020 The Apache Software Foundation

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

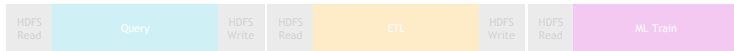
利用事例：Amazon Athena



利用事例：RAPIDS

Data Processing Evolution Faster Data Access, Less Data Movement

Hadoop Processing, Reading from Disk



Spark In-Memory Processing



25-100x Improvement
Less Code
Language Flexible
Primarily In-Memory

Traditional GPU Processing



5-10x Improvement
More Code
Language Rigid
Substantially on GPU

RAPIDS



50-100x Improvement
Same Code
Language Flexible
Primarily on GPU

<https://docs.rapids.ai/overview/RAPIDS%200.15%20Release%20Deck.pdf#page=8>

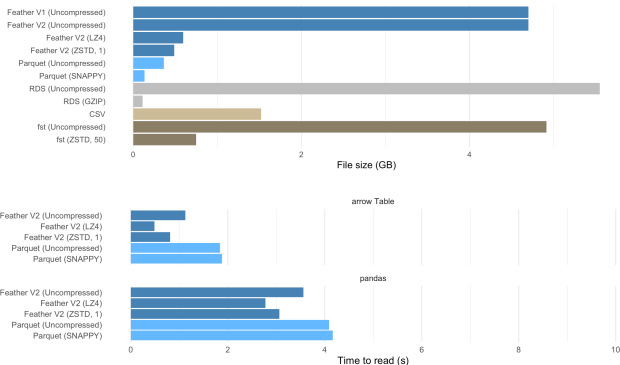
データサイズは？

- ✓ 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}シリアルライズが速い
 - ✓ 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・I/O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

列				Apache Arrow	RDBMS	列			
	a	b	c			a	b	c	
1	値	値	値		1	値	値	値	
2	値	値	値		2	値	値	値	
3	値	値	値		3	値	値	値	
列ごと				値の管理単位	行ごと				
列				高速なアクセス単位	行				

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

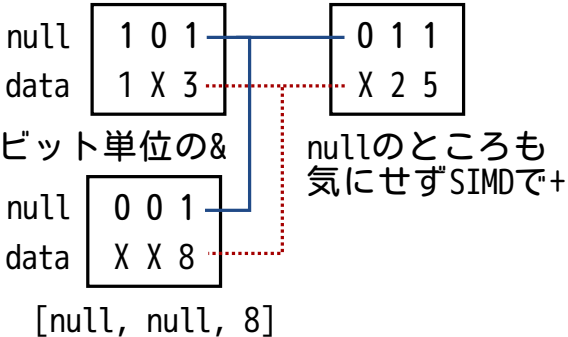
定数時間でi番目にアクセスできない

nullと条件分岐

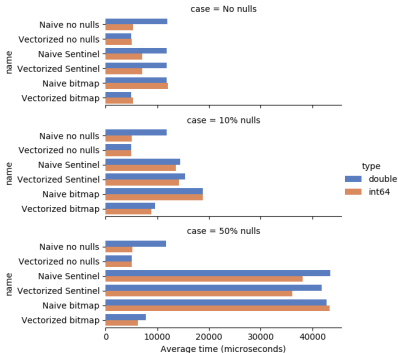
- ✓ null対応のアプローチ：
 - ✓ null値：Julia: missing, R: NA
 - ✓ 別途ビットマップを用意：Apache Arrow
- ✓ ビットマップを使うと条件分岐をなくせる

nullと条件分岐とSIMD

$[1, \text{null}, 3] + [\text{null}, 2, 5]$



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ーマットが適切
- ✓ 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
 - ✓ なんかデータ分析に便利そうなすごいやつ！
 - ✓ 「よくわからん」と言われるからといって全体像を説明してもらえなかった！
 - ✓ 使い方も説明してもらえなかった！
- ✓ Apache Arrowフォーマット
 - ✓ これだけ説明してもらえた！

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・I/O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・I/O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！！！！

次回予告！

- ✓ 案1：Apache Arrowデータの高速処理
- ✓ 案2：Apache Arrowデータの高速RPC
- ✓ 案3：〇〇言語でApache Arrowを使う方法
- ✓ ...

次のステップ

- ✓ もっと詳しく知りたくなったから
イベント・社内・…で紹介して！
 - ✓ <https://www.clear-code.com/contact/>
- ✓ 使いたくなったから技術サポートして！
 - ✓ <https://www.clear-code.com/contact/>
- ✓ Apache Arrowの開発に参加したい！
 - ✓ <https://arrow.apache.org/community/>