Apache Arrow Based Workflows for Large Data Analytics

Kyle Baron

kyleb@metrumrg.com

Outline

- Motivating remarks
- Performance working with single files
    - Delimited text
    - Serialized object
    - Fast formats for rectangular data
- Parquet data format
- Apache arrow
    - Single file API
    - Data set API
- Arrow examples in R

Resources

- Big Data in R with Arrow
  - 1-Day Posit::Conf (2023) Workshop
  - https://posit-conf-2023.github.io/arrow/
- Larger-Than-Memory Data Workflows with Apache Arrow
  - 2022 UseR! Conference
  - https://arrow-user2022.netlify.app/
- Apache Arrow documentation
  - https://arrow.apache.org/docs/
  - https://arrow.apache.org/docs/r/
- Feather V2 with Compression Support in Apache Arrow 0.17.0
  - https://ursalabs.org/blog/2020-feather-v2/
- Arrow Cheatsheet
  - https://github.com/apache/arrow/blob/main/r/cheatsheet/arrow-cheatsheet.pdf
- Materials for this talk
  - https://github.com/kylebaron/data-2024

Resources

- Apache Arrow Overview
  - https://arrow.apache.org/overview/
- François Michonneau Blog
  - https://francoismichonneau.net/2022/08/arrow-dataset-creation/
  - https://francoismichonneau.net/2022/09/arrow-dataset-part-2/
  - https://francoismichonneau.net/2022/10/import-big-csv/

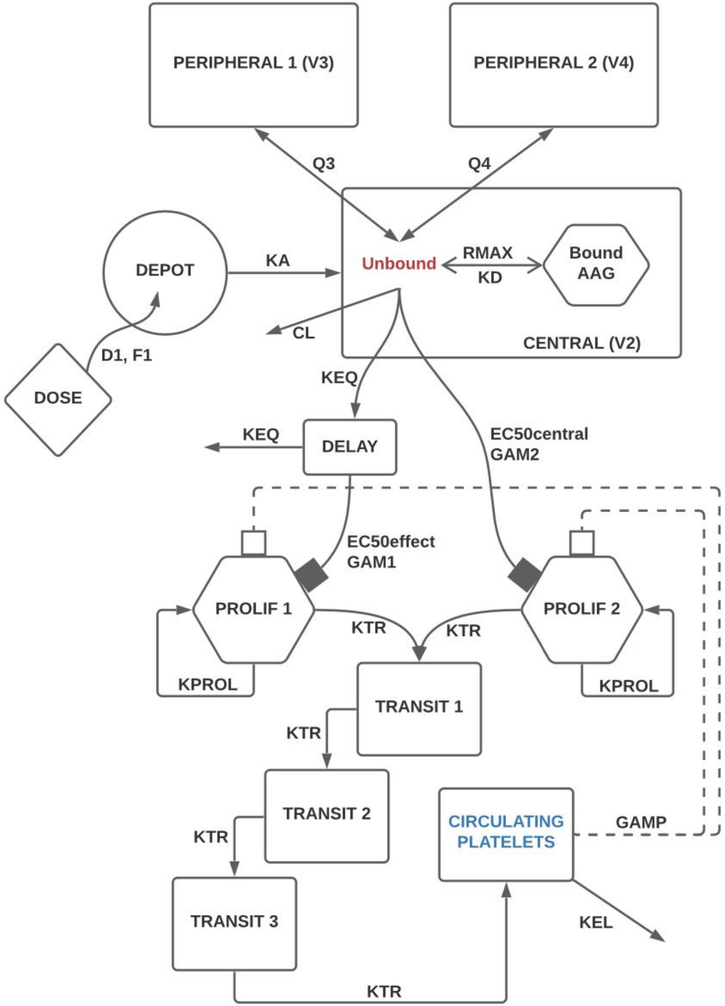
Where I've been

College of Pharmacy

University of Minnesota
Driven to Discover®

College of Pharmacy
Experimental and Clinical Pharmacology

METRUM
RESEARCH GROUP



So, what do you mean by "Big Data"?

UnitedHealth Group
Minnetonka, MN

Fortune Global 500 list of 2023

Rank	Company	Country	Industry	Revenue in USD
1	Walmart	United States	Retail	$611.3 billion
2	Saudi Aramco	Saudi Arabia	Energy	$603.7 billion
3	State Grid	China	Energy	$530.0 billion
4	Amazon	United States	Internet services and retailing	$514.0 billion
5	China National Petroleum	China	Petroleum	$483.1 billion
6	Sinopec Group	China	Petroleum	$471.2 billion
7	ExxonMobil	United States	Petroleum	$413.7 billion
8	Apple	United States	Technology	$394.3 billion
9	Shell	United Kingdom	Petroleum	$386.2 billion
10	UnitedHealth Group	United States	Health care	$324.2 billion

I don't usually work with billions of records

```
> library(data.table)

> data <- fread("pk-data.csv")
   user  system elapsed
  0.343   0.000   0.100

> dim(data)
[1] 59,845    76

> file.size("pk-data.csv")/1e6 # MB
[1] 24.86339
```

Really?

Simulation output from a recent project

```
data/sim$ du -sh pd-sim*.parquet

902M    pd-sim-out-1-tn1.parquet
902M    pd-sim-out-1-tn2.parquet
902M    pd-sim-out-1-tn3.parquet
902M    pd-sim-out-1-tn4.parquet
902M    pd-sim-out-1-tn5.parquet
```

```
> data <- lapply(files, arrow::read
   user  system elapsed
  3.496   4.468   1.619

> sum(sapply(data, nrow))
[1] 195,000,000
```

"Big Data" can factor into your analyses in a variety of ways

Finding the right approach to handling bulky data can greatly streamline your analyses

- Read efficiency
- Write efficiency
- Storage efficiency
- Analysis efficiency

Working with single files

data.ext ⟶ **R data frame**

Delimited text		
utils	**read.csv(), write.csv()**	What you used when you learned R
readr	**read_csv(), write_csv()**	Lots of use; "tidy"
vroom	**vroom(), vroom_write()**	Lazy load via ALT REP; R 3.5+ only
data.table	**fread(), fwrite()**	Known as the speed champ
arrow	**read_csv_arrow(), write_csv_arrow()**	Is it faster than fread()?
Serialized object		
base	**readRDS(), saveRDS()**	Comes with R; slow; handles any object
qs	**qread(), qwrite()**	My favorite; doesn't deal with stan objects well
Rectangular, fast		
fst	**read_fst(), write_fst()**	Replaced by qs; part of the "fastverse"
arrow	**read_parquet(), write_parquet()**	Wide use
arrow	**read_feather(), write_feather()**	New and fast

NYC Taxi Data (tiny)

- 1,672,513 rows
- 24 columns
- Multiple data types
  - Integer
  - Numeric
  - Character
  - Datetime

```
library(arrow)
bucket <- s3_bucket("voltrondata-labs-datasets/nyc-taxi-tiny")
copy_files(from = bucket, to = "nyc-taxi")
```

Read times (ms) - NYC taxi data set

	Function	Time	Relative	Min	Max	Data	n
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<int>
1	utils::read.csv()	9169	5.18	8240	10840	taxi	10
2	readr::read_csv()	1771	1	1596	2247	taxi	10
3	data.table::fread()	732	0.413	655	847	taxi	10
4	vroom::vroom()	541	0.305	458	1045	taxi	10
5	arrow::read_csv_arrow()	280	0.158	248	355	taxi	10
6	base::readRDS()	1703	0.962	1525	2752	taxi	10
7	qs::qread()	387	0.218	351	430	taxi	10
8	fst::read_fst()	466	0.263	322	1099	taxi	10
9	arrow::read_feather()	132	0.0748	32	656	taxi	10
10	arrow::read_parquet()	98	0.0553	41	180	taxi	10

Write times (ms) - NYC taxi data set

	Function	Time	Relative	Min	Max	Data	n
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<int>
1	utils::write.csv()	16568	3.45	15072	17903	taxi	10
2	readr::write_csv()	4797	1	3980	6145	taxi	10
3	data.table::fwrite()	984	0.205	975	1007	taxi	10
4	vroom::vroom_write()	4760	0.992	3847	6859	taxi	10
5	arrow::write_csv_arrow()	5958	1.24	5908	5996	taxi	10
6	base::saveRDS()	9458	1.97	9409	9555	taxi	10
7	qs::qsave()	505	0.105	477	639	taxi	10
8	fst::write_fst()	311	0.0648	290	356	taxi	10
9	arrow::write_feather()	201	0.0419	191	212	taxi	10
10	arrow::write_parquet()	652	0.136	634	686	taxi	10

File size - NYC taxi data set

	Format	Size	Unit	Relative	Data
	<chr>	<dbl>	<chr>	<dbl>	<chr>
1	csv	236.	MB	1	taxi
2	rds	51.3	MB	0.218	taxi
3	qs	60.1	MB	0.255	taxi
4	fst	114.	MB	0.484	taxi
5	parquet	54.4	MB	0.231	taxi
6	feather	111.	MB	0.470	taxi

Data compression - NYC taxi data set

	Format	Comp	Size	Unit	Relative	Data
	<chr>	<chr>	<dbl>	<chr>	<dbl>	<chr>
1	csv	no	236.	MB	1	taxi
2	feather	yes	111.	MB	0.470	taxi
3	parquet	yes	54.4	MB	0.231	taxi
4	feather	no	296.	MB	1.25	taxi
5	parquet	no	59.6	MB	0.253	taxi

Parquet format

- Columnar storage
- Compression by column
- Bit packing
- Store row group stats
- Dictionary encoding
- Run-length encoding

Row-Based Storage

1,Marc,Johnson,Dallas,27

2,Ja

3,Ja

Colu

ID:

FIRS

LAST ley

CITY: Dallas,Denver,Chicago

AGE: 27,35,51

https://www.upsolver.com/blog/apache-parquet-why-use

Parquet Encodings

Product	Quantity	OrderDate
Product A	1	06/12/2021 19:01:15.000
Product A	1	07/12/2021 19:01:16.000
Product B	1	08/12/2021 19:01:16.231
Product C	2	09/12/2021 19:01:17.000
Product A	1	10/12/2021 19:01:18.000
Product B	1	11/12/2021 19:01:19.565
Product B	2	12/12/2021 19:01:20.000
Product A	2	13/12/2021 19:01:20.876
Product A	2	14/12/2021 19:01:21.500
Product C	1	15/12/2021 19:01:22.000

Product
Product A
Product A
Product B
Product C
Product A
Product B
Product B
Product A
Product A
Product C

Product
0
0
1
2
0
1
1
0
0
2

Dictionary (Product)	
0	Product A
1	Product B
2	Product C

Dictionary encoding

Quantity
1
1
1
2
1
1
3
3
3
3

Run length encoding

Quantity (RLE)
3
1
1
2
2
1
4
3

https://dataninjago.com/2021/12/07/databricks-deep-dive-3-parquet-encoding/

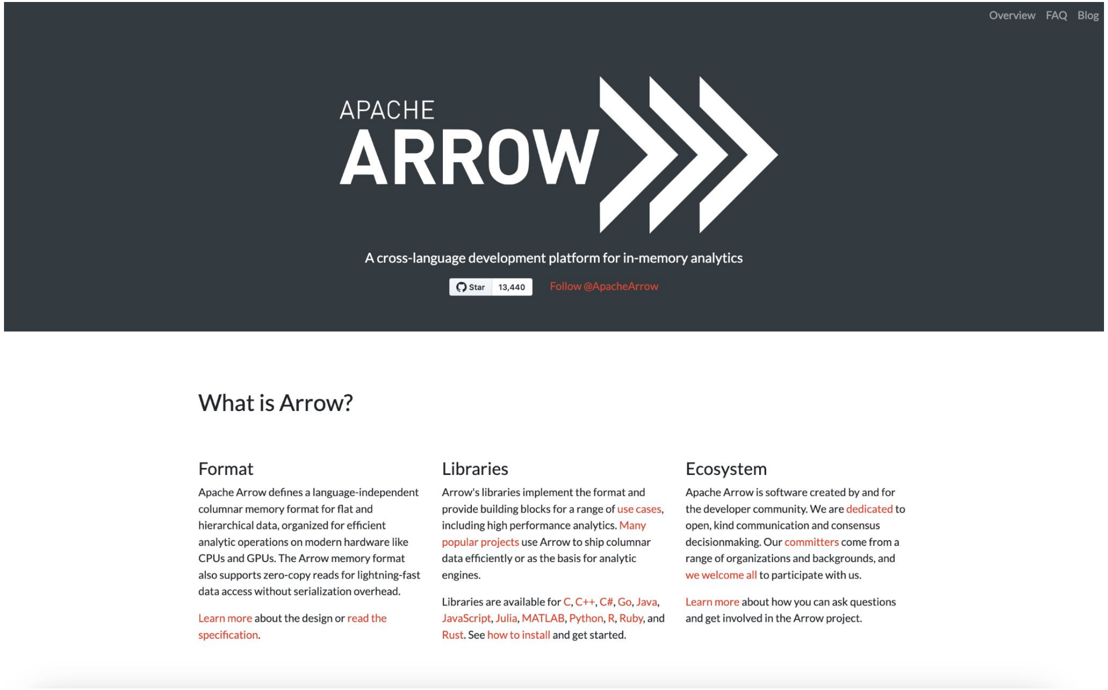
Data compression and encoding - NYC taxi data set

	Format	Comp	Dict	Size	Unit	Relative	Data
	<chr>	<chr>	<chr>	<dbl>	<chr>	<dbl>	<chr>
1	parquet	yes	yes	54.4	MB	1	taxi
2	parquet	no	yes	59.6	MB	1.10	taxi
3	parquet	yes	no	88.4	MB	1.63	taxi
4	parquet	no	no	242.	MB	4.46	taxi

https://arrow.apache.org/docs/r/articles/install.html

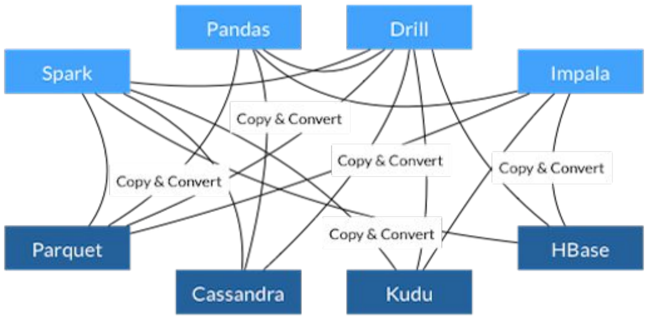
- Several performant options for reading csv
- `fwrite()` was fastest csv writer
- `qread() / qwrite()` good all around
- Arrow
  - feather and parquet are both very fast
  - parquet has great storage
  - parquet is special

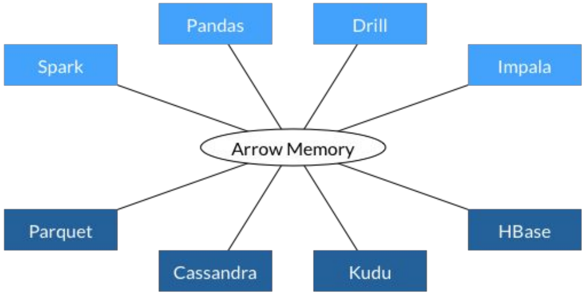
What is Apache Arrow?



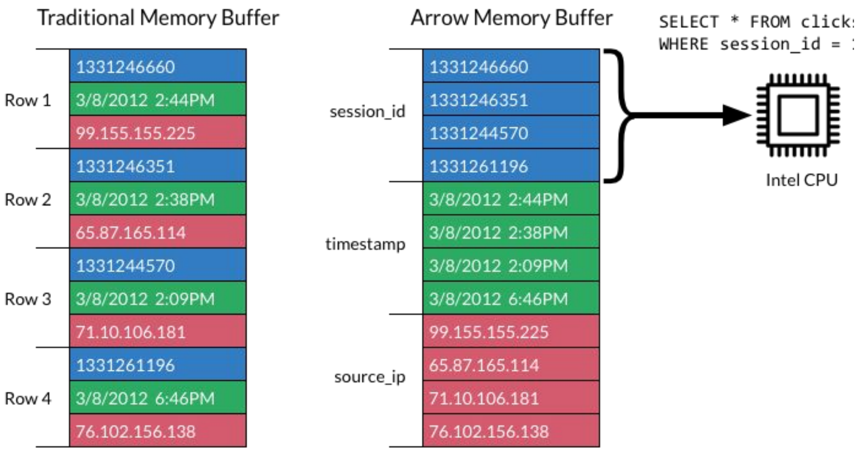
- Cross-language
- Development platform
- In-memory analytics
- Not (just) a *file* format

Standardized, language-agnostic in-memory format





	session_id	timestamp	source_ip
Row 1	1331246660	3/8/2012 2:44PM	99.155.155.225
Row 2	1331246351	3/8/2012 2:38PM	65.87.165.114
Row 3	1331244570	3/8/2012 2:09PM	71.10.106.181
Row 4	1331261196	3/8/2012 6:46PM	76.102.156.138

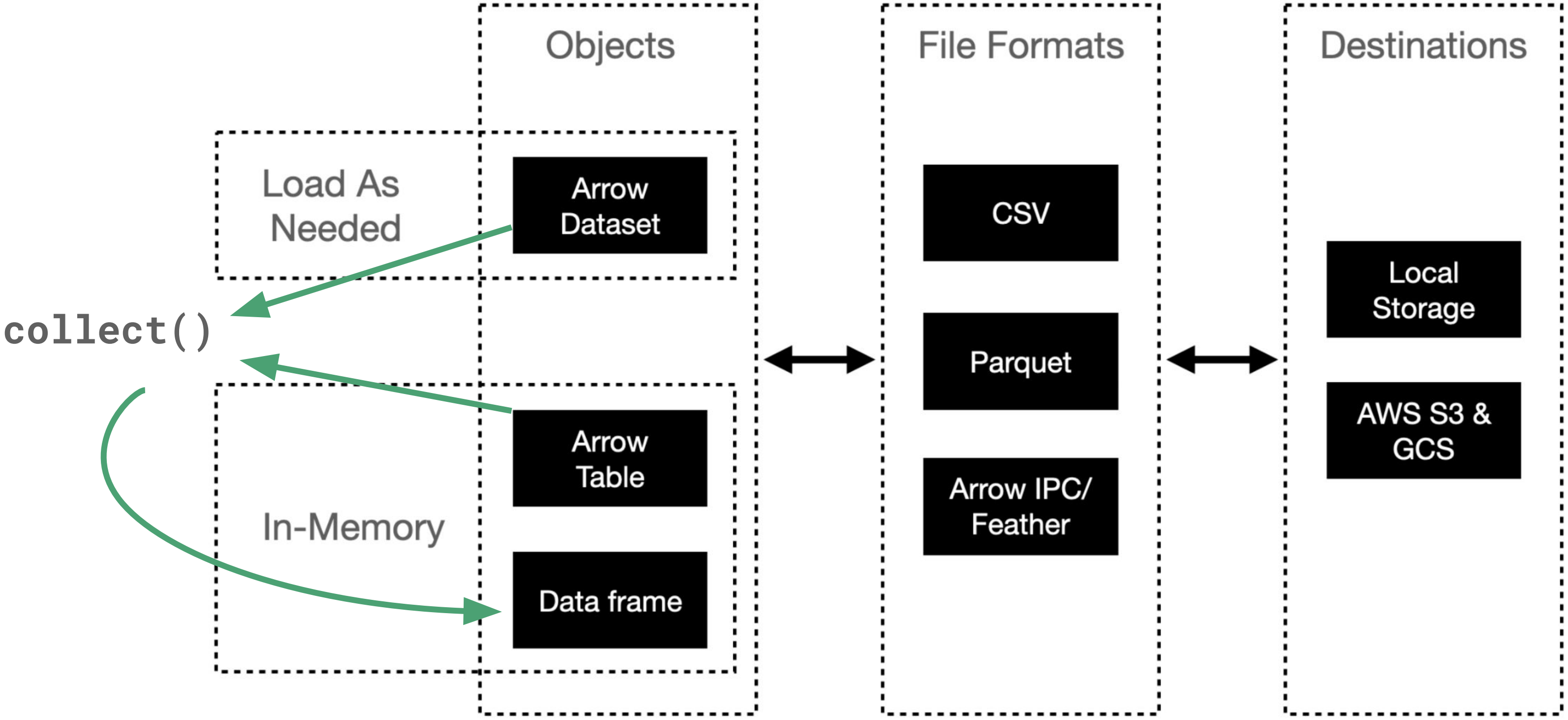


https://arrow.apache.org/overview/

Arrow APIs

- Single-file API
  - `arrow::read_parquet(), arrow::write_parquet()`
  - Data is stored in a (single) file
  - Read can return data.frame or arrow table
  - Data is in memory
- Dataset API
  - `arrow::open_dataset(), arrow::write_dataset()`
  - Data is stored a directory
    - One or more files
    - Directory can be partitioned in subdirectories
    - Data engineering
  - Data is loaded only when needed

Arrow APIs



https://posit-conf-2023.github.io/arrow/materials/1_hello_arrow.html#/arrow-1

Single-file and data set API

Read a single file, returns data frame in memory

```
df <- read_parquet("big-data.parquet")
```

Read a single file, returns arrow table in memory

```
at <- read_parquet("big-data.parquet", as_data_frame = FALSE)

at <- arrow_table(df)
```

Read directory, returns data set object, instant

```
ds <- open_dataset("data-dir/")
class(ds)
[1] "FileSystemDataset" "Dataset" "ArrowObject" "R6"
```

"tidy" workflow for arrow data sets and tables

Arrow table (at): in memory
Arrow data set (ds): on disk (memory-mapped)
Both: return R data frame via **`collect()`**

```
at %>%
  group_by(var1, var2) %>%
  summarise(Mean = mean(value)) %>%
  collect()

ds %>%
  group_by(var1, var2) %>%
  summarise(Mean = mean(value)) %>%
  collect()
```

Apache Arrow Demo

- Simulate from compartmental PK model
  - 4 dose levels
  - 300 subjects per dose
  - 26 output records per subject
  - 3000 replicates
- Output
  - 3.6 million subjects
    - Random variability
  - 9.36 million rows, 12 columns
  - 3.3 GB (parquet); 7 GB (feather)
- All simulation and analysis done on Apple M1 Pro (2021) 16 GB