Taming the language border in data analytics and science with Apache Arrow

Uwe Korn – QuantCo – 18th June 2019

quantco

About me

- Engineering at QuantCo
- Apache {Arrow, Parquet} PMC
- Focus on Python but interact with R, Java, SAS, …

@xhochy
@xhochy
mail@uwekorn.com
https://uwekorn.com

Do we have a problem?

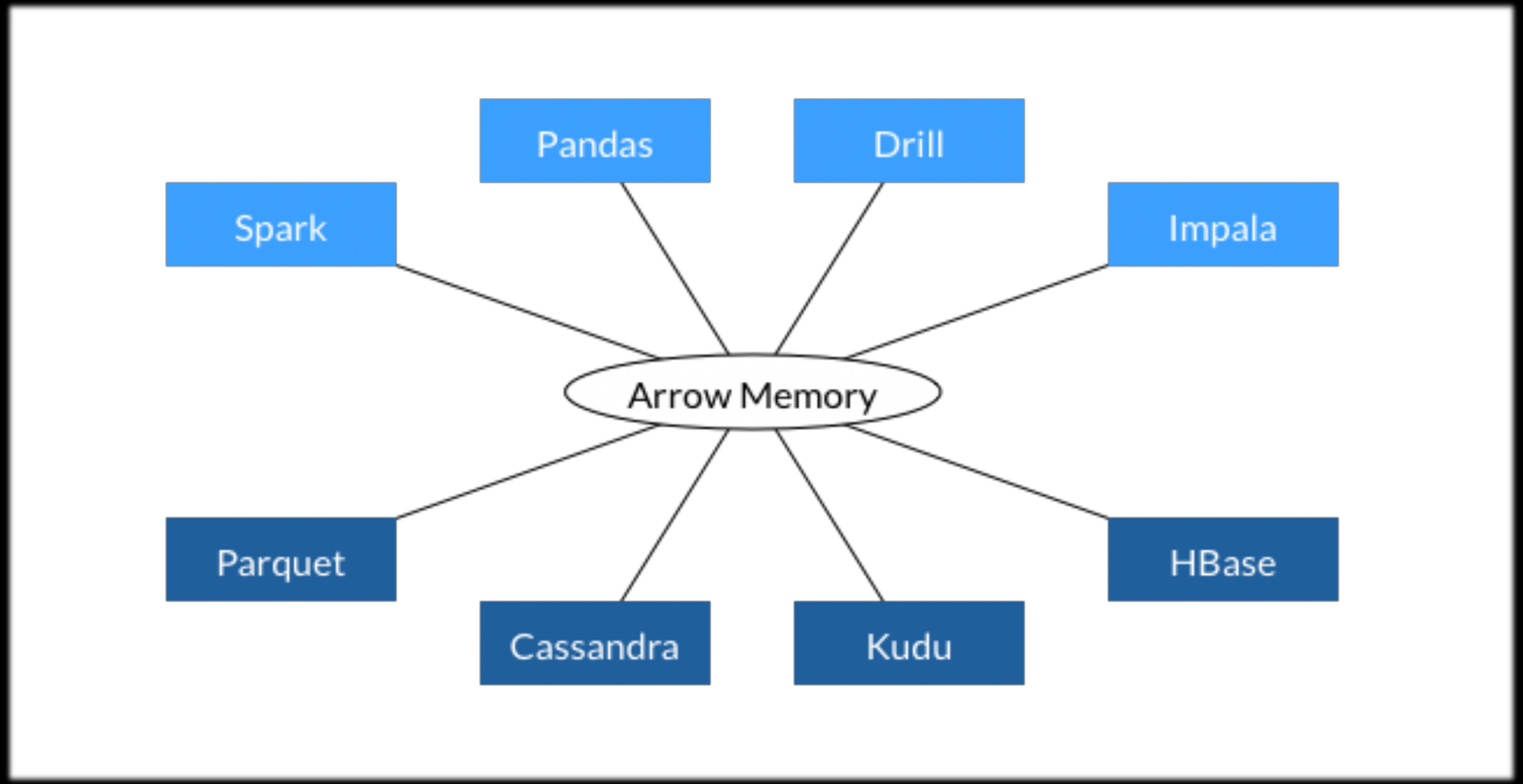
- Yes, there are different ecosystems!
- Berlin Buzzwords
  - Java / Scala
  - Flink / ElasticSearch / Kafka
  - Scala-Spark / Kubernetes
- PyData
  - Python / R
  - Pandas / NumPy / PySpark/sparklyr / Docker
- SQL-based databases
  - ODBC / JDBC
  - Custom protocols (e.g. Postgres)

quantco

Why solve this?

- We build pipelines to move data
- Goal: end-to-end data products
  Somewhere along the path we need to talk
- Avoid duplicate work / work on converters

quantco


Pandas
Drill
Spark
Impala
Arrow Memory
Parquet
HBase
Cassandra
Kudu

quantco

Apache Arrow at its core

- **Main idea:** *common columnar representation of data in memory*
- Provide libraries to access the data structures
- Broad support for many languages
- Create building blocks to form an ecosystem around it
- Implement adaptors for existing structures

quantco

	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

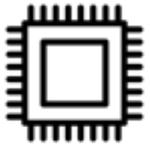
Traditional Memory Buffer

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

Arrow Memory Buffer

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

```
SELECT * FROM clickstream
WHERE session_id = 1331246351
```



Intel CPU

quantco

Previous Work

- CSV works really everywhere
  - Slow, untyped and row-wise
- Parquet is gaining traction in all ecosystems
  - one of the major features and interaction points of Arrow
  - Still, this serializes data
- RAM-Copy: 10GB/s on a Laptop
- DataFrame implementations look similar but still are incompatible

quantco

Languages

- C++, C(glib), Python, Ruby, R, Matlab
- C#
- Go
- Java
- JavaScript
- Rust

quantco

There’s a social component

- It’s not only APIs you need to bring together
- Communities are also quite distinct
- Get them talking!

quantco

Shipped with batteries

- There is more than just *data structures*
- Batteries in Arrow
  - Vectorized Parquet reader: C++, Rust, Java(WIP)
    C++ also supports ORC
  - Gandiva: LLVM-based expression kernels
  - Plasma: Shared-memory object store
  - DataFusion: Rust-based query engine
  - Flight: RPC protocol built on top of gRPC with zero-copy optimizations

quantco

Ecosystem

- RAPIDS: Analytics on the GPU
- Dremio: Data platform
- Turbodbc: columnar ODBC access in C++/Python
- Spark: fast Python and R bridge
- fletcher (pandas): Use Arrow instead of NumPy as backing storage
- fletcher (FPGA): Use Arrow on FPGAs
- Many more … https://arrow.apache.org/powered_by/

quantco

Does it work?

Everything is amazing on slides …

… so does this Arrow actually work?

Let’s take a real example with:

- ERP System in Java with JDBC access (no non-Java client)
- ETL and Data Cleaning in Python
- Analysis in R

quantco

Does it work?

```
import jpype
import pyarrow as pa

batch = jpype.JPackage("org").apache.arrow.adapter.jdbc.JdbcToArrow.sqlToArrow(
    connection,
    query,
    ra
)

record_batch = pyarrow.jvm.record_batch(batch)
```

quantco

Does it work?

```
df = record_batch.to_pandas()
df.info()
# <class 'pandas.core.frame.DataFrame'>
# RangeIndex: 3 entries, 0 to 2
# Data columns (total 2 columns):
# id       3 non-null int64
# price    3 non-null float64
# dtypes: float64(1), int64(1)
# memory usage: 128.0 bytes
```

```
print(df)
#       id  price
# 0  1001   1.04
# 1  1002   2.50
# 2  1003   3.99
```

quantco

Does it work?

```r
library(reticulate)
pa <- import("pyarrow")
py_table <- some$python$code
py_table
# pyarrow.Table
# id: int64
# price: double
table <- py_to_r(py_table)
as.data.frame(table)
#     id price
# 1 1001  1.04
# 2 1002  2.50
# 3 1003  3.99
```

WIP

quantco

Up Next

- Build more adaptors, e.g. Postgres
- Building blocks for query engines on top of Arrow
  - Datasets
  - Analytical kernels
- DataFrame implementations directly on top of Arrow

quantco

Thanks

Slides at https://twitter.com/xhochy
Question here!

quantco