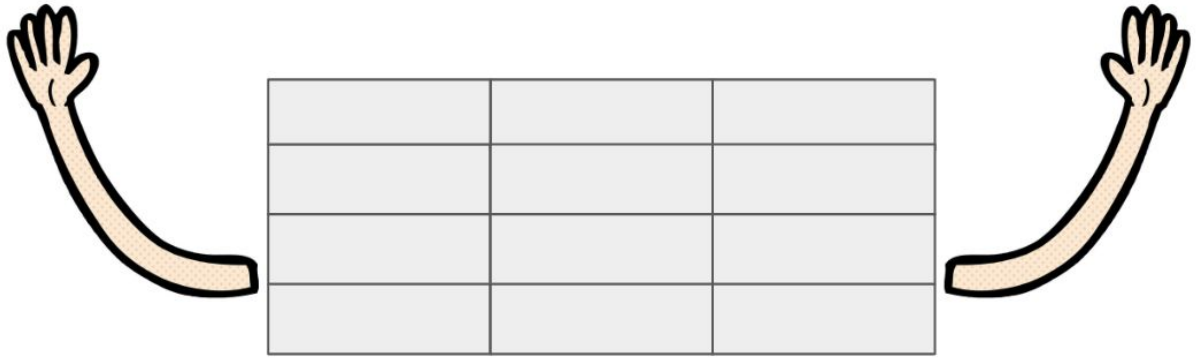


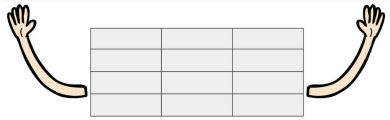
AKIMBO

When your data doesn't fit in your dataframe

Martin Durant, Anaconda

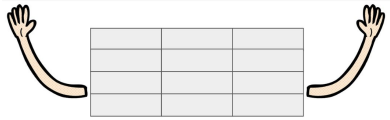
PyData Global

ANACONDA.



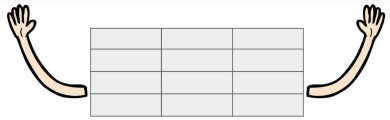
Outline

- Motivation
  - Physics origins
- Simple demo
- More demo
- Data sources
- Current status and plans



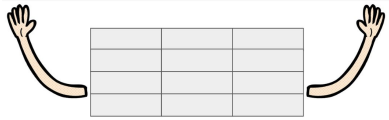
Motivation: dataframes

- Tabular data is:
  - Group of one-dimensional columns
  - One data type per column
  - Core to SQL
- Tables are usually “dataframes” in python
  - Pandas
  - Polars
  - Pyspark
  - many others
- Dataframes offer:
  - Vectorized processing in-memory or out-of-core
  - SQL-like methods (join, unique...)



Motivation: arrow

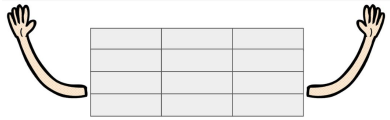
- **Python object iteration (e.g., JSON) is slow and memory-heavy**
- Parquet data model is more general
  - Missing data (OPTIONAL)
  - Variable-length types (LIST)
  - Nested records (STRUCT)
  - Call this "JSON-like" data
- The `arrow` data model fully implements JSON-like data
- More recent binary data formats target `arrow`
- `arrow` backs most dataframes and tabular data interchange



Motivation: akimbo

- Awkward-array (`ak`) is a `numpy`-like API over JSON-like data, built for physics
- Implements vectorized kernels (fast c++)
- Works on CPU and GPU
- Integrates with Numba (CPU/GPU)

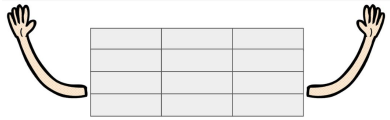
Akimbo integrated the `ak` API into your favourite dataframe library as an accessor, for the "awkward" parts of your data



Akimbo

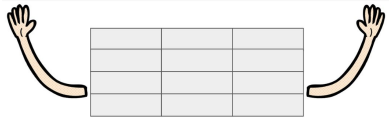
Akimbo integrates with Pandas, Polars, dask-dataframe, and cuDF, through an **.ak** accessor.

- NumPy-like API
- apply (u)funcs and aggregations on any level
- string and datetime ops in nested structures
- GPU/CPU Numba support
- attach OOP-like behaviours to struct fields and data-specific operations



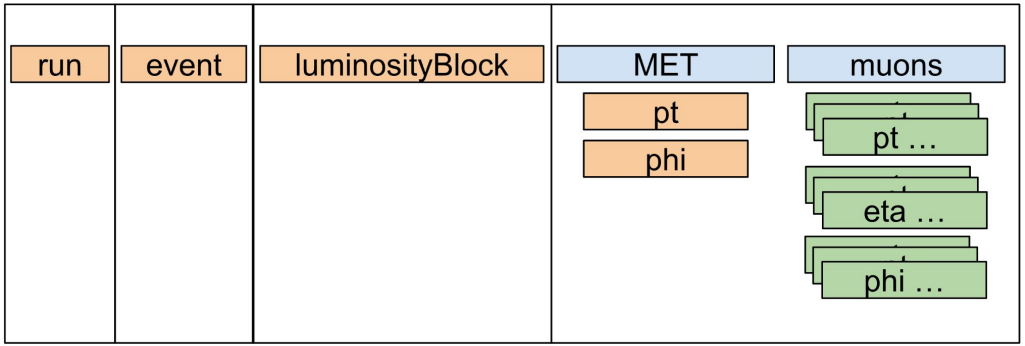
Akimbo

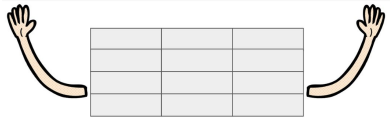
```
s = pd.Series([[1, 2, 3], [0], [4, 5]] * 100000)

s + 1                   # doesn't work
s.ak + 1                # [[2, 3, 4], [1], …]

s.ak.max(axis=1)    # [3, 0, 5, …]

s.ak.apply(numba_func)
```



Akimbo

- Quickstart: https://akimbo.readthedocs.io/en/latest/quickstart.html
- Detailed HEP walkthrough: https://github.com/intake/akimbo/blob/main/docs/demo/akimbo-demo.ipynb



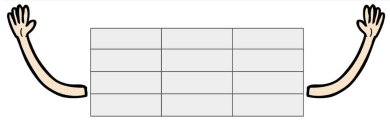


Akimbo - usage DEMO

PyData Global

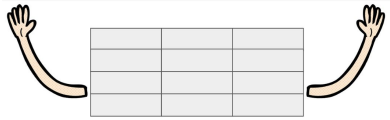
Martin Durant

ANACONDA.



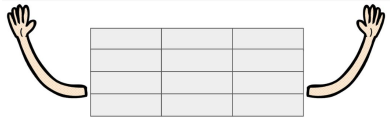
Features - recap

- Numpy API
- (u)funcs, operator overload, broadcasting
- General apply and transform
- Use numba function (CPU/GPU)
- str, dt and pluggable subaccessors
- “Objects” behaviour definition



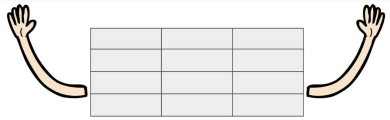
Features - recap

- Pandas
- Polars (eager)
- Dask.dataframe (intra-partition)
- cuDF
- (daft, pyspark)



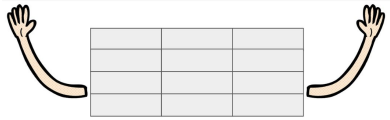
Akimbo - multi DEMO

Martin Durant



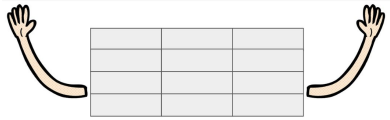
Data Sources

- Binary columnar files (parquet, feather, lance ...)
- Text files (JSON, XML, logfiles ...)
- Binary record files (avro, msgpack, ...)
- One-to-many joins (ORM)
- Exotic data formats (sensors, ...)
- ... ?



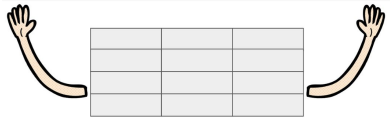
Data Sources - BYO

- We want workflows!
- Example: `nested-pandas` showed a dataframe-in-dataframe workflow (each row is a time-series), based on two-table merge
- Example types:
  - Vectors
  - Anything geometric
  - IPs (`akimbo-ip`)



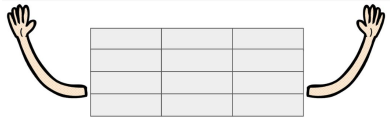
Discussion points

- Why not just SQL?
- What about file-backed SQL (e.g., duckdb)?
- When is iteration enough?
- When is explode() enough?
- How to integrate with expr/lazy ?
  - ak provides the typetracer
  - Dask-awkward already knows which columns to read
- Can someone make a better LOGO?



Plans

- More backends
- More types
- More examples
- Feedback!



Resources

Presentation material:
https://github.com/martindurant/pydata_global_2024

Docs: https://akimbo.readthedocs.io/

Code: https://github.com/intake/akimbo

Awkward docs: https://awkward-array.org/

PyData Global

ANACONDA.