11-2018

Open Data Standards for Administrative Data Processing

Ryan M. White
Statistics Canada

DOI https://doi.org/10.23889/ijpds.v3i5.1045

Open Data Standards for Administrative Data Processing

Abstract
Adoption of non-traditional data sources to augment or replace traditional survey vehicles can reduce respondent burden, provide more timely information for policy makers, and gain insights into the society that may otherwise be hidden or missed through traditional survey vehicles. The use of non-traditional data sources imposes several technological challenges due to the volume, velocity and quality of the data. The lack of applied industry-standard data format is a limiting factor which affects the reception, processing and analysis of these data sources. The adoption of a standardized, cross-language, in-memory data format that is organized for efficient analytic operations on modern hardware as a system of record for all administrative data sources has several implications:

- Enables the efficient use of computational resources related to I/O, processing and storage.
- Improves data sharing, management and governance capabilities.
- Increases analyst accessibility to tools, technologies and methods.

Statistics Canada developed a framework for selecting computing architecture models for efficient data processing based on benchmark data pipelines representative of common administrative data processes. The data pipelines demonstrate the benefits of a standardized data format for data management, and the efficient use of computational resources. The data pipelines define the preprocessing requirements, data ingestion, data conversion, and metadata modeling, for integration into a common computing architecture. The integration of a standardized data format into a distributed data processing framework based on container technologies is discussed as a general technique to process large volumes of administrative data.

Comments
DOI https://doi.org/10.23889/ijpds.v3i5.1045

Open Data Standards for Administrative Data Processing

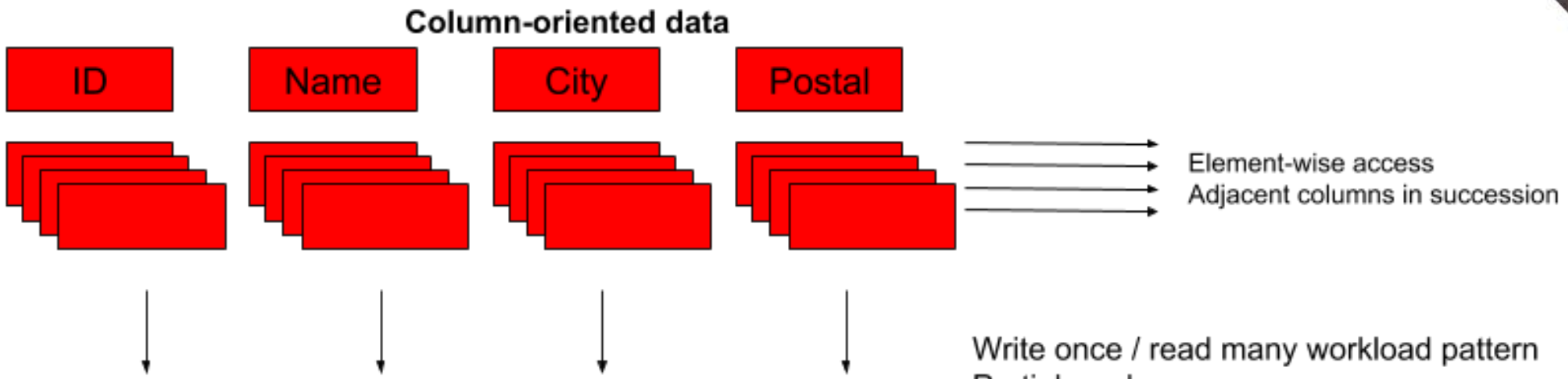
Ryan M White, PhD

2018 ADRF Network Research Conference
Washington, DC, USA
November 13th to 14th, 2018

Features & Challenges

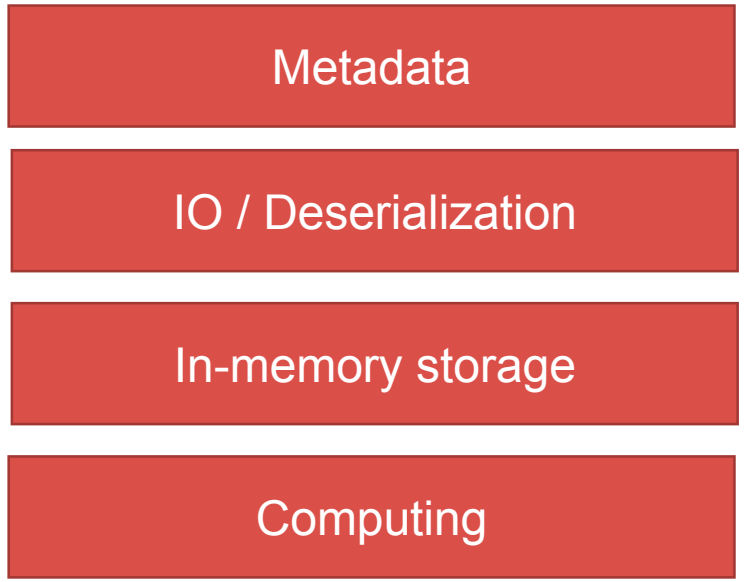
- Preservation of the raw state of the data is essential to extract relevant and accurate statistics
- External data sources with lack of common data standard or format
- Analytical workloads and data access patterns differ from traditional survey data

Administrative Data Analytical Workloads



What if?

- Administrative data can be represented as a table
  - Preserves the raw state with a rich data format
- Standard tabular data format common to all systems
  - Common data interface!
  - Zero-copy data exchange between systems, processes, libraries
- Systems are modular



Open Data Standards

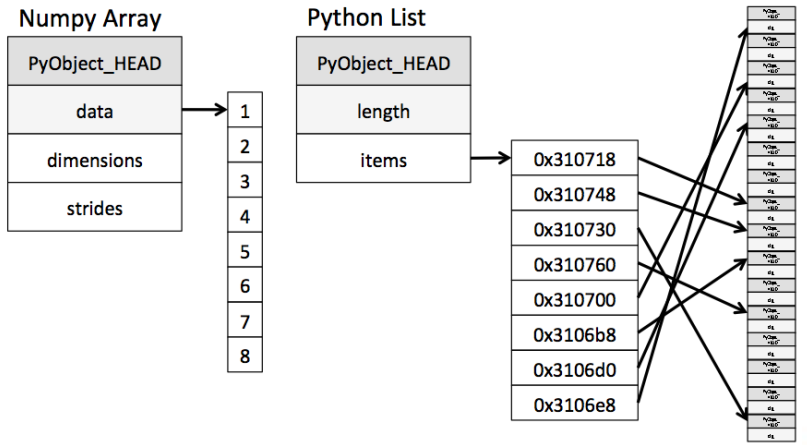
- Open standards enables systems to directly communicate
  - Simplifies system architecture
  - Reduces ecosystem fragmentation
  - Improves interoperability across processes
  - Reduces dependencies on proprietary systems and data formats
- Common data formats encourage and facilitate collaboration
  - Facilitates code-reuse supported by large communities
  - Simplifies data management, data sharing and data access

Common Data Formats

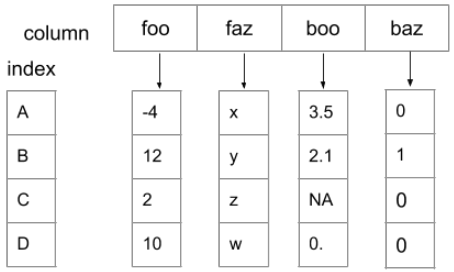
- Examples of data standards in computing today:
  - Human-readable semi-structured: XML, JSON
  - Structure data query language: SQL has various flavors (MySQL, PostgreSQL, etc...)
  - Binary data (with metadata) (BigData & scientific community): NetCDF, HDF5, Apache Parquet, ORC, ROOT
  - Binary blobs Serialization / RPC protocols: Apache Avro, Protocol buffers (Google)
- Non-open data formats
  - CSV, Excel, SAS bdat

Standard in-memory data

- Scientific community converged around strided ndarray (NumPy) $n_{offset} = \sum_{k=0}^{N-1} s_k n_k$
  - Fortran compatible (column-major) $s_k^{column} = itemsize \prod_{j=0}^{k-1} d_j$
  - C compatible (row-major) $s_k^{row} = itemsize \prod_{j=k+1}^{N-1} d_j$
- Benefits
  - Contiguous in-memory (i.e., single-segment, memory layouts, access any part of memory block via indices)
  - Zero-overhead memory sharing between libraries in-memory and processes via shared memory
  - Algorithm re-use – scientific libraries developed in Fortran
  - Reuse IO and storage



Tabular data already exists!



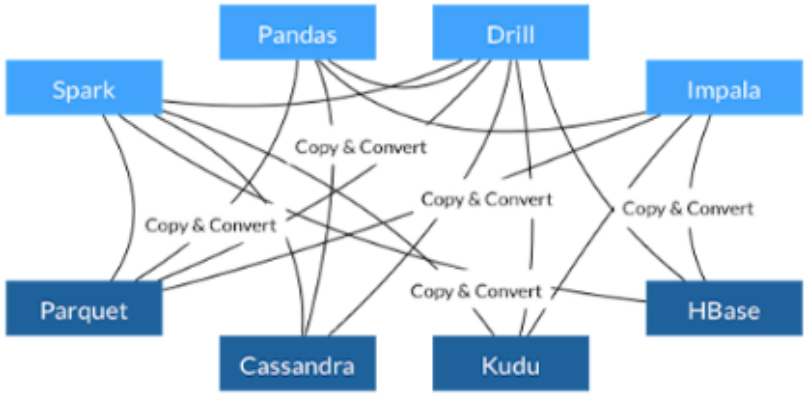
- Tabular data ubiquitous in Data and Social Sciences.
  - Notoriously not based on open standards
- *DataFrame* (table*)* concept and semantics are nearly identical across various systems.
  - In-memory representation varies dramatically.
  - Supported data types / structures varies.
  - Algorithmic code across languages cannot be shared/re-used.
- Examples of *DataFrames*
  - Tabular Data is common in SQL
  - Big data systems developed Spark and Hive
  - Popular data science languages all have a *DataFrame* (R, python, Julia)

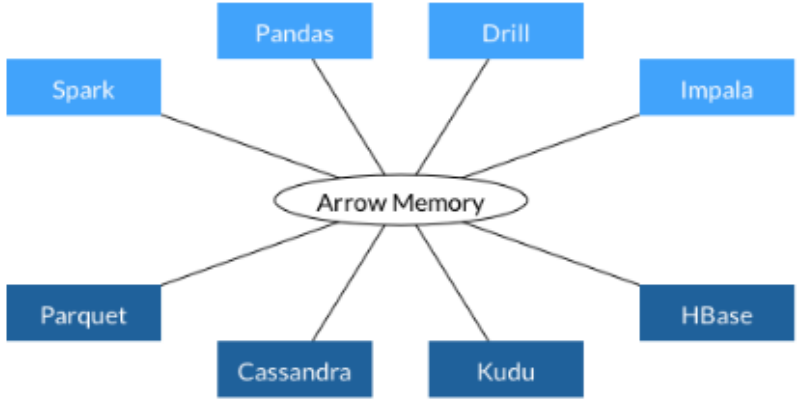
Apache Arrow

- Cross-language development platform for in-memory data
  - Solves the non-portable *DataFrame* problem
- Standard
  - Specifies language-independent columnar memory format for flat and hierarchal data
  - Backed by key developers in major open-source projects
- Fast
  - Data locality
  - Zero-copy
- Flexible
  - Common data structure
  - Interface across systems and languages
- Sustainable
  - Algorithms, tools and libraries can be re-used
  - Modular, deconstructed data architecture

Apache Arrow – Common Data Layer

Advantages of a Common Data Layer





- Each system has its own internal memory format
- 70-80% computation wasted on serialization and deserialization
- Similar functionality implemented in multiple projects

- All systems utilize the same memory format
- No overhead for cross-system communication
- Projects can share functionality (eg, Parquet-to-Arrow reader)

StatCan Artemis

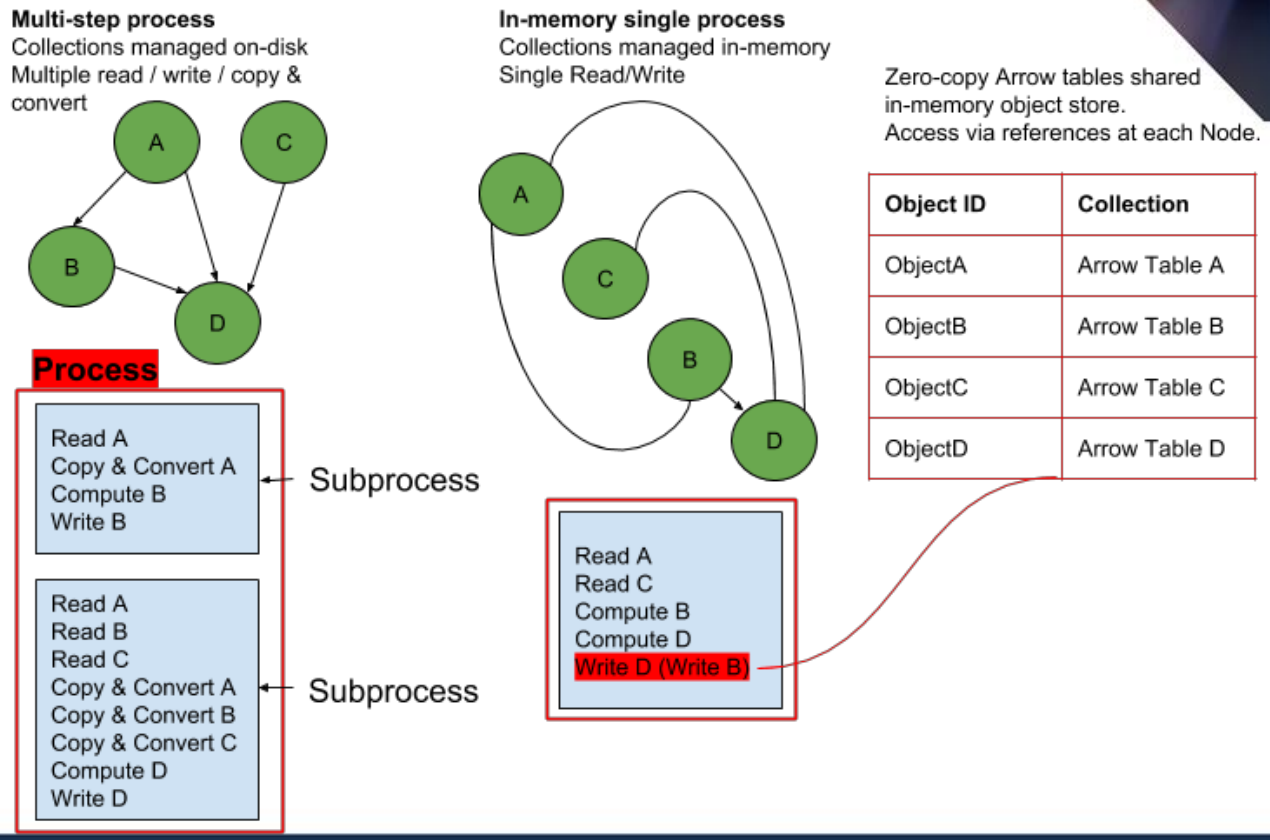
- Prototype framework for processing Administrative Data powered by Apache Arrow.
- Assumptions to test
  - Data converted, stored and analyzed in a tabular data format.
  - Data partitioned (datums/batches), processed independently.
  - Multiple business processes represented as Directed Graphs (DAGs) applied to each partition in-memory.

Artemis Prototype Design Features

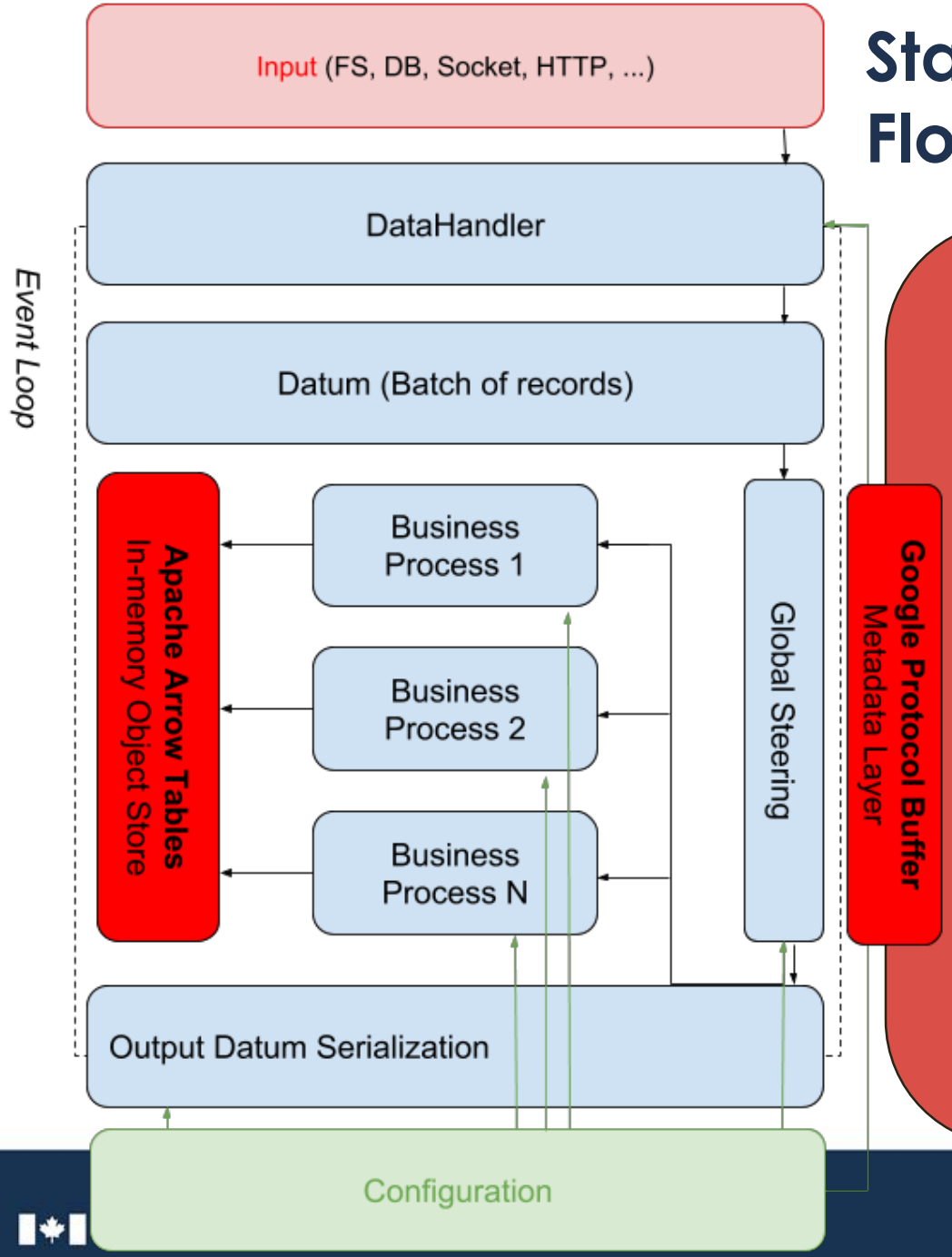
- Metadata management - separation of algorithmic code and configuration
- Performance - separation of I/O and data pipelining
- Reliability – job-level control flow and global steering algorithm for data pipelines
- Reproducibility - in-memory provenance of data transformations
- Flexibility
  - Modular code design to facilitate code re-use, testing and development
  - User-defined algorithms, tools, histograms and data tables
  - Data access provides Arrow tables to user-defined components
- Real-time data monitoring, validation and statistical analysis
  - Histogram-based monitoring, cost analysis and data profiling

In-memory Data Provenance and Data Access

- Directed Graph of all Business Process known at configuration.
  - Define inputs, algorithms and output
- Topological sort of DAG is a linear ordering of the vertices
  - Execution order which ensures data dependencies are met.
- DAG data structure in memory
  - Create anchors at each vertex, retains references to intermediate data produced during processing.
  - Data access (references) of Arrow Tables (object store) to user-defined algorithms
  - Collection of objects and serialization can be managed by framework (collections to serialize from objet store)



StatCan Artemis Control Flow & Metadata



Metadata Model

Process Configuration

- Execution graph
- DAG structure
- Algorithm configuration

Job Configuration

- Data input
- Glob pattern
- Data Chunking

Job process

- Payload information
- Data provenance
- Data schema
- Histograms
- Timers

Summary

- **Open data standards well-established in scientific community**
  - Preserve, share and re-use knowledge from a large community
- **NSOs, social and data scientists analyze tabular data**
  - Our systems do not communicate effectively
- **Apache Arrow™ is a cross-language development platform**
  - Single data format and standard will allow for data exchange
  - Simplifies system architecture and data management
  - Deconstruct the typical data architecture stack
  - Facilitate collaboration
  - Drive growth and innovation
- **StatCan Artemis project powered by Apache Arrow™**
  - Experimental data processing and analysis framework for the efficient use of Administrative Data.

THANK YOU! MERCI!

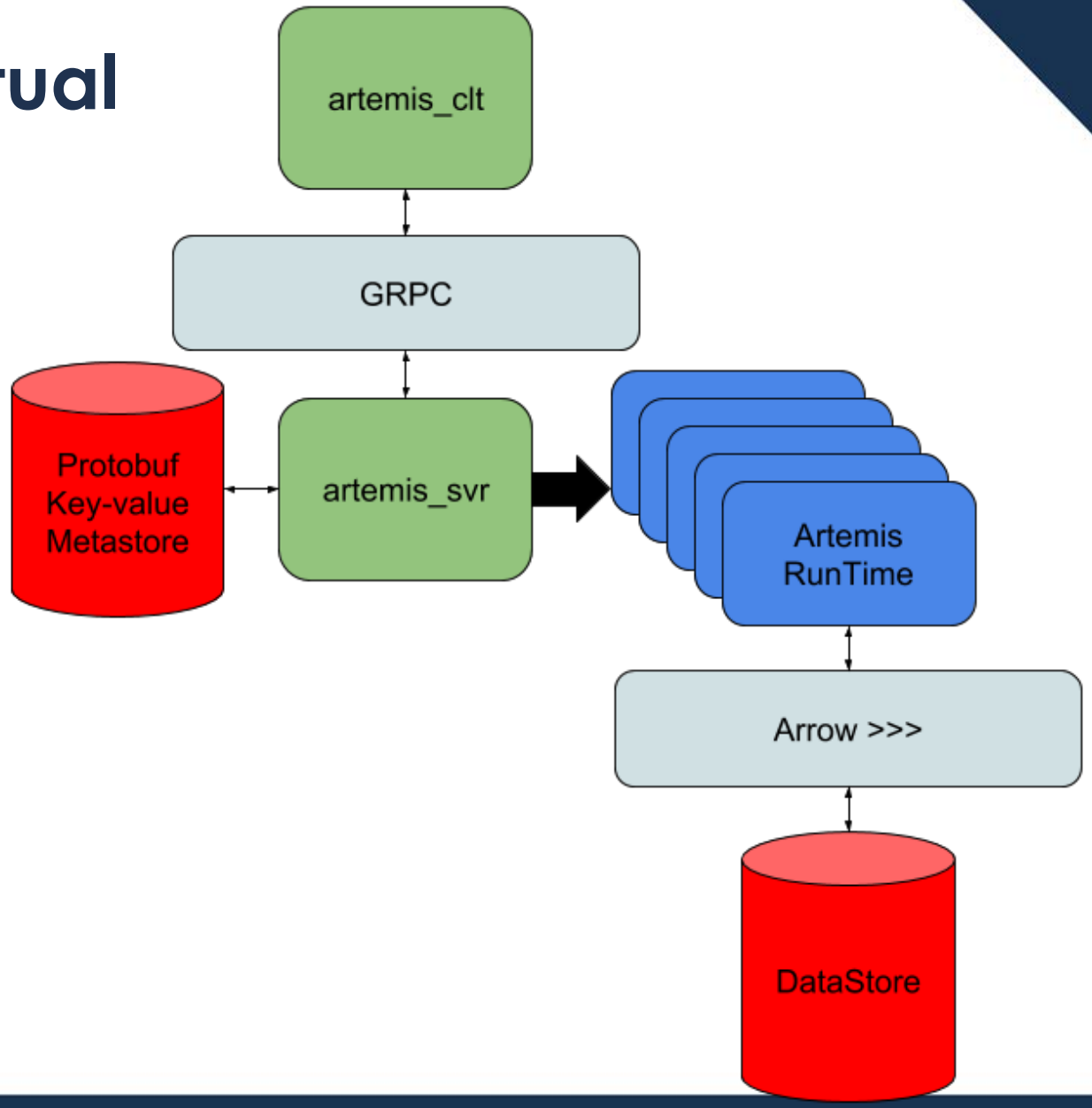
For more information please visit,
www.statcan.gc.ca

#StatCan100

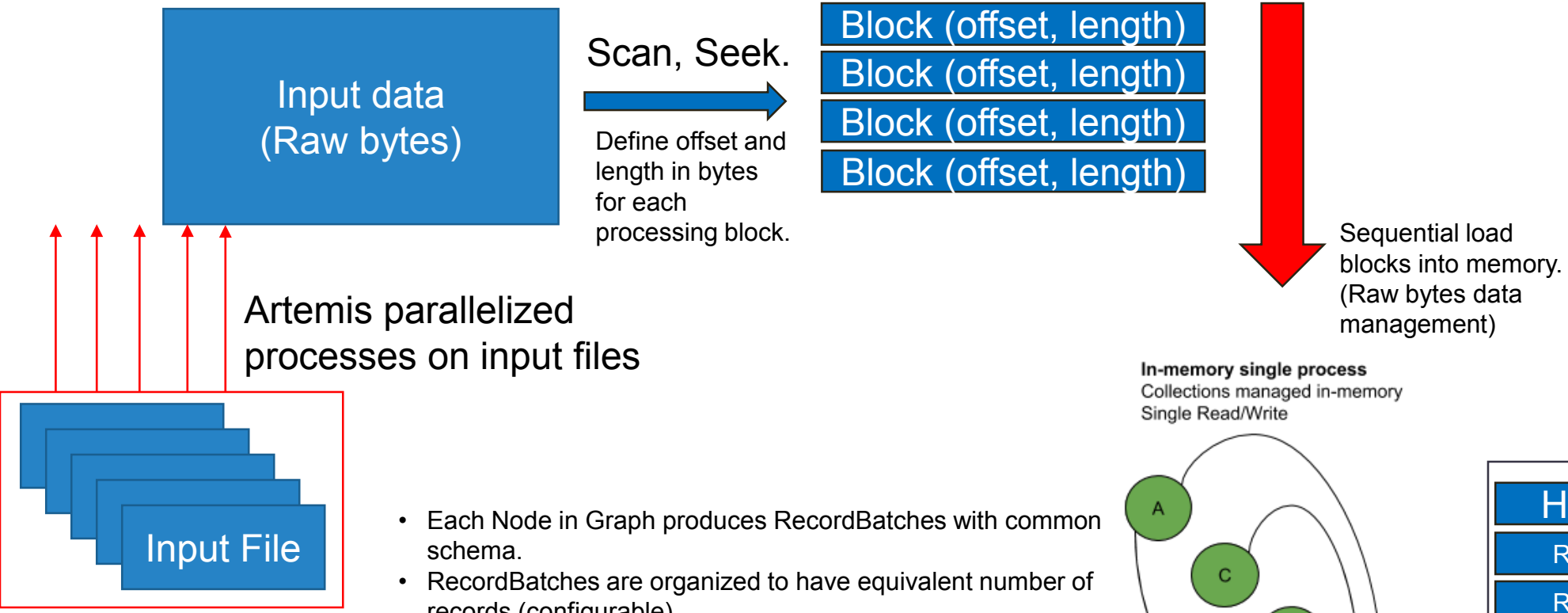
References

- *The NumPy array: a structure for efficient numerical computation*, Van Der Walt, Stefan; Colbert, S. Chris; Varoquaux, Gaël, Computing in Science and Engineering 13, 2 (2011) 22-30, 2011arXiv1102.1523V
- *Why Python is Slow*, https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/
- Apache Arrow™, https://arrow.apache.org/
- Google Protocol Buffers, https://developers.google.com/protocol-buffers/

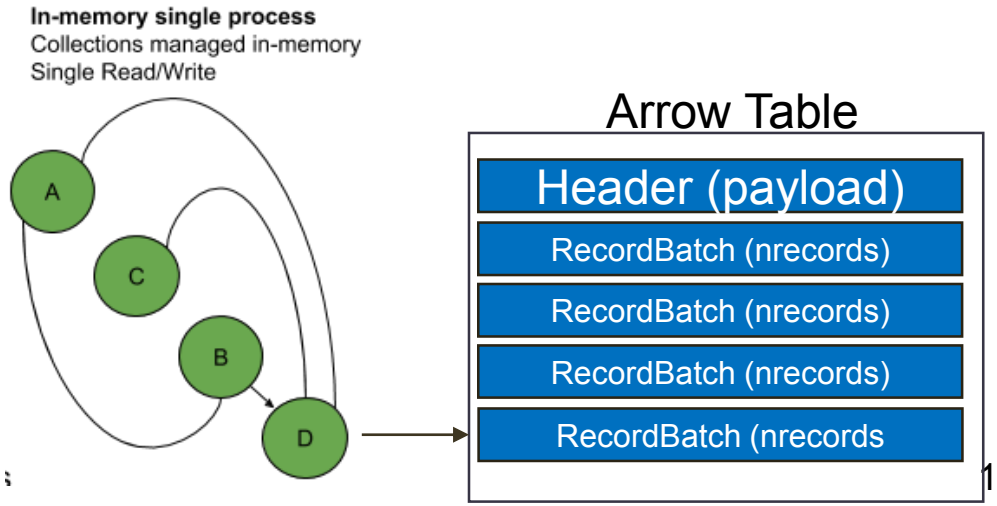
Artemis Conceptual Architecture



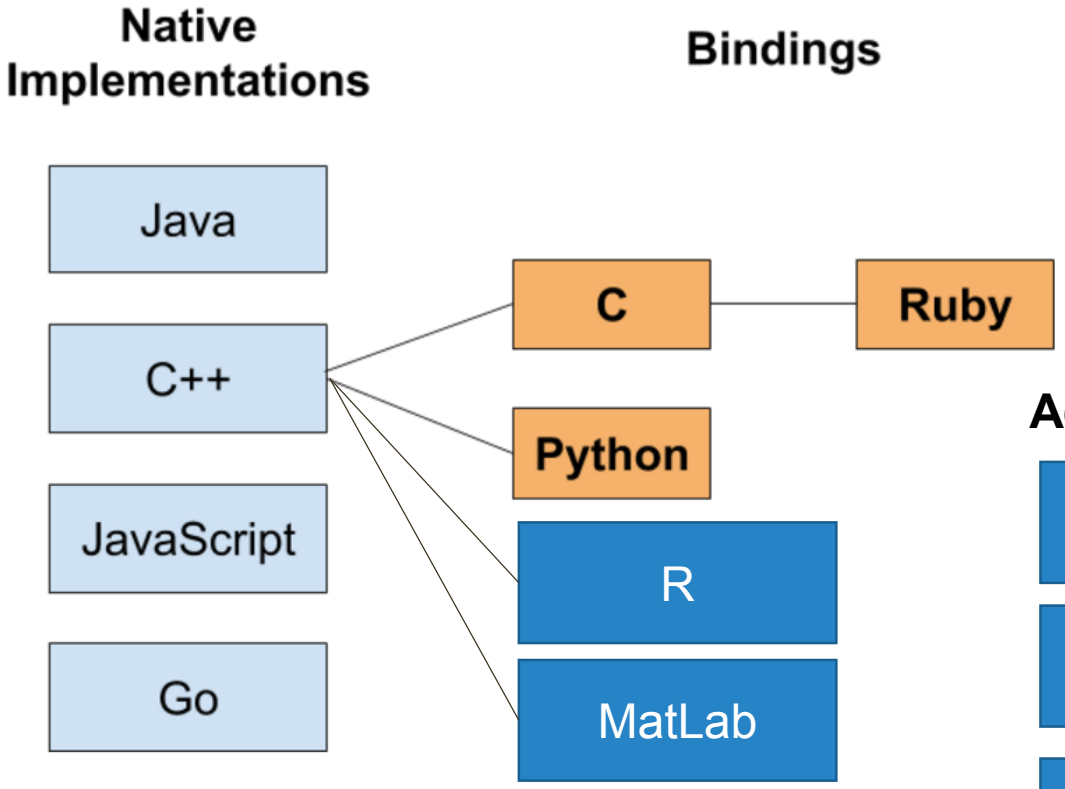
Artemis Core Data Flow



- Each Node in Graph produces RecordBatches with common schema.
- RecordBatches are organized to have equivalent number of records (configurable).
- RecordBatches are combined to form Arrow Tables.
- Arrow Tables are serialized and written to disk.
- Data provenance is retained in metadata (Protobufs).
- Tables are flushed to disk to have optimal data file size.



Apache Arrow Cross-Platform



Additional contributions for cross-platform support

Rust

.NET

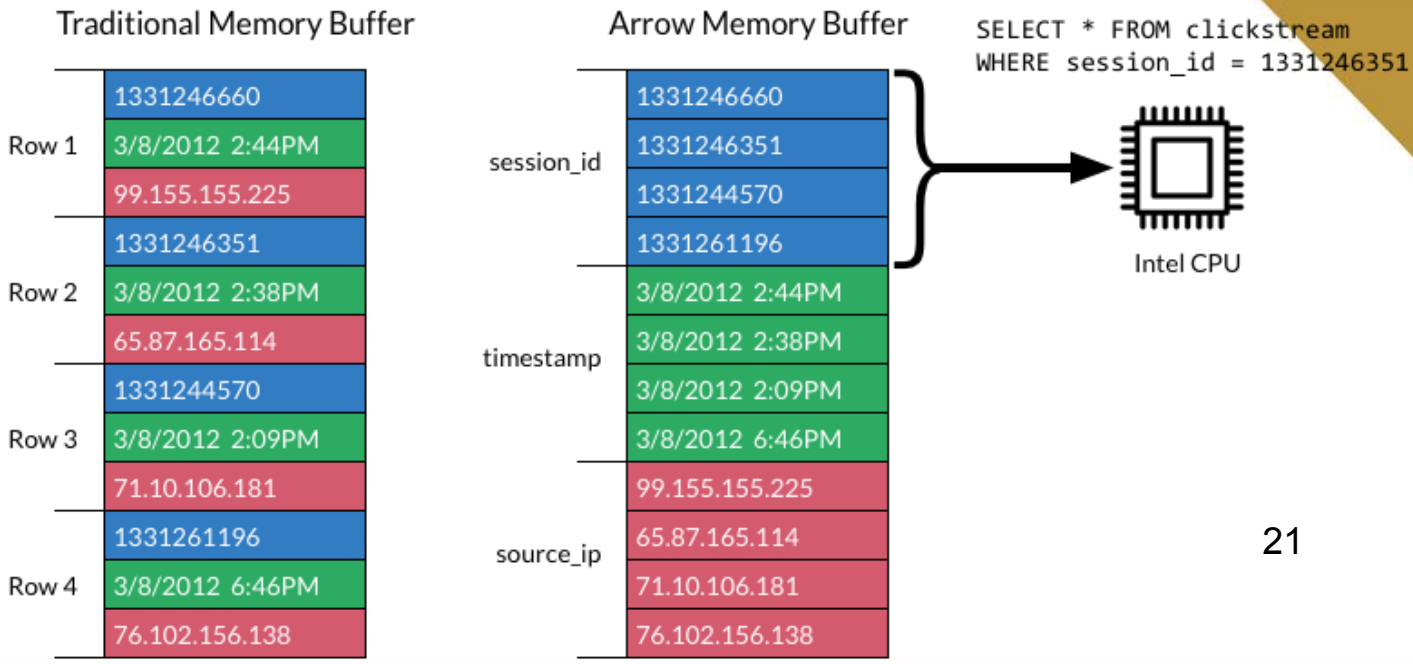
CUDA → Pyarrow/Numba CUDA Interop!

Rapid adoption of Arrow for expanding interoperability. Democratizing system development and data access.

Arrow Focus on CPU Efficiency

- Maximize CPU throughput
  - SIMD (Single instruction multiple data)
  - Cache locality
- Vectorized operations
- Minimal structure overhead (only up-front cost)
- Efficient, lightweight compression schema per column

	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



Arrow Data Format

- Scalars – supports fixed-width and variable-width
  - Boolean
  - [u]int[8,16,32,64], Decimal, Float, Double
  - Date, Time, Timestamp
  - UTF8 String, Binary
- Complex
  - Struct, List, Union
- Validity buffer bit-map retaining null value position in column
- https://cwiki.apache.org/confluence/display/ARROW/Columnar+Format+1.0+Milestone

Notable Developments

- Arrow Flight RPC and Messaging Framework.
  - gRPC Arrow-based messaging.
  - Avoid expensive parsing when receiving data over the wire.
- Parquet and Arrow C++ converge to single repo / dev cycle.
- C++ CSV Reader Object (early performance studies with Artemis)
- R Library development
- CUDA-based GPUs in Python. Interop with Numba and GPU Open Analytics Initiative.