

The future of column-oriented data processing with Arrow and Parquet

Julien Le Dem, Principal Architect Dremio, VP Apache Parquet




HELLO
MY NAME
IS



Julien Le Dem

@J_



- Architect at @DremioHQ  **dremio**
- Formerly Tech Lead at Twitter on Data Platforms.
- Creator of Parquet
- Apache member
- Apache PMCs: Arrow, Incubator, Pig, Parquet



Agenda

- Community Driven Standard
- Interoperability and Ecosystem
- Benefits of Columnar representation
 - On disk (Apache Parquet)
 - In memory (Apache Arrow)
- Future of columnar



Community Driven Standard



An open source standard

- Parquet: Common need for on disk columnar.
- Arrow: Common need for in memory columnar.
- Arrow building on the success of Parquet.
- Benefits:
 - Share the effort
 - Create an ecosystem
- Standard from the start



The Apache Arrow Project

- New Top-level Apache Software Foundation project
 - Announced Feb 17, 2016
- Focused on Columnar In-Memory Analytics
 1. 10-100x speedup on many workloads
 2. Common data layer enables companies to choose best of breed systems
 3. Designed to work with any programming language
 4. Support for both relational and complex data as-is
- Developers from 13+ major open source projects involved
 - A significant % of the world's data will be processed through Arrow!

Calcite

Cassandra

Deeplearning4j

Drill

Hadoop

HBase

Ibis

Impala

Kudu

Pandas

Parquet

Phoenix

Spark

Storm

R



Interoperability and Ecosystem



Shared Need => Open Source Opportunity

“We are also considering switching to a columnar canonical in-memory format for data that needs to be materialized during query processing, in order to take advantage of SIMD instructions” -Impala Team

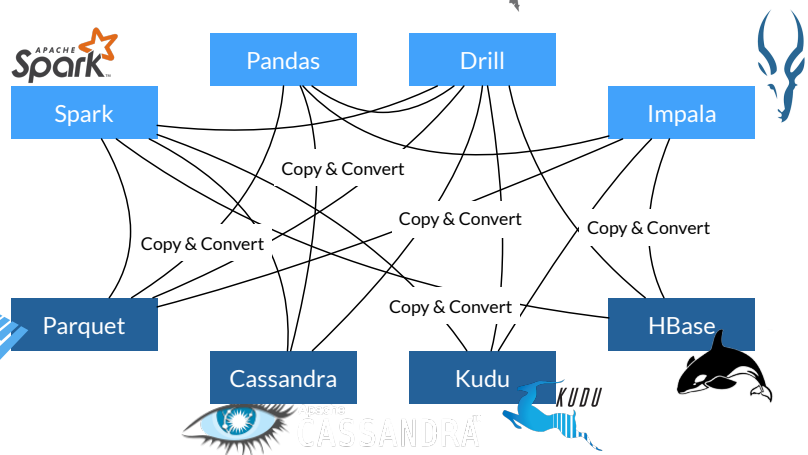
“Drill provides a flexible hierarchical columnar data model that can represent complex, highly dynamic and evolving data models and allows efficient processing of it without need to flatten or materialize.” -Drill Team

“A large fraction of the CPU time is spent waiting for data to be fetched from main memory...we are designing cache-friendly algorithms and data structures so Spark applications will spend less time waiting to fetch data from memory and more time doing useful work” – Spark Team



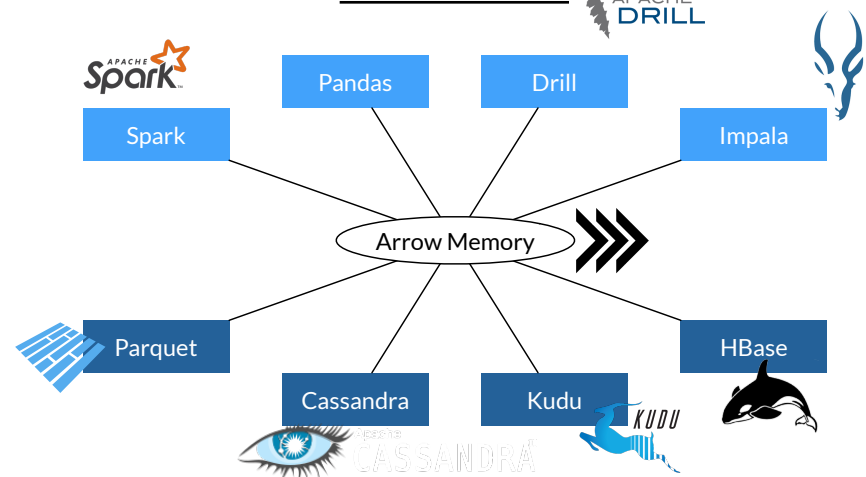
High Performance Sharing & Interchange

Before



- Each system has its own internal memory format
- 70-80% CPU wasted on serialization and deserialization
- Functionality duplication and unnecessary conversions

With Arrow



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



Benefits of Columnar formats



@EmergencyKittens



Columnar layout

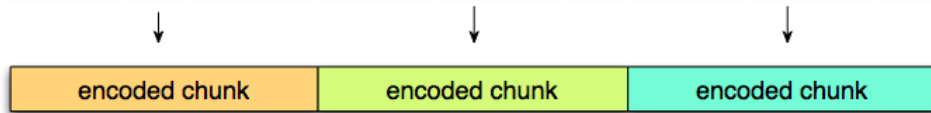
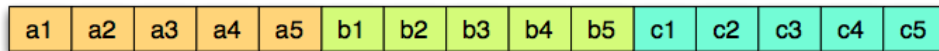
Logical table representation

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

Row layout



Column layout



On Disk and in Memory

- Different trade offs
 - On disk: Storage.
 - Accessed by multiple queries.
 - Priority to I/O reduction (but still needs good CPU throughput).
 - Mostly Streaming access.
 - In memory: Transient.
 - Specific to one query execution.
 - Priority to CPU throughput (but still needs good I/O).
 - Streaming and Random access.



Parquet on disk columnar format



Parquet on disk columnar format

- Nested data structures
- Compact format:
 - type aware encodings
 - better compression
- Optimized I/O:
 - Projection push down (column pruning)
 - Predicate push down (filters based on stats)



Access only the data you need

Columnar

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

Statistics

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

+

=

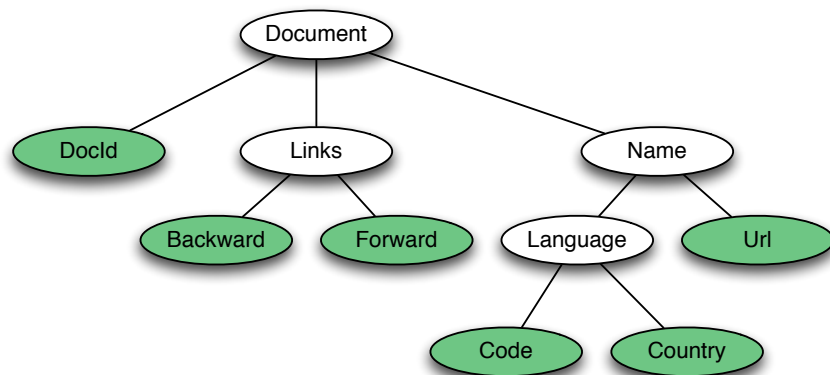
Read only the data you need!

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5



Parquet nested representation

Borrowed from the Google Dremel paper



Columns:

docid
links.backward
links.forward
name.language.code
name.language.country
name.url

<https://blog.twitter.com/2013/dremel-made-simple-with-parquet>



Arrow in memory columnar format



Arrow goals

- Well-documented and cross language compatible
- Designed to take advantage of modern CPU characteristics
- Embeddable in execution engines, storage layers, etc.
- Interoperable

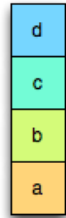


Arrow in memory columnar format

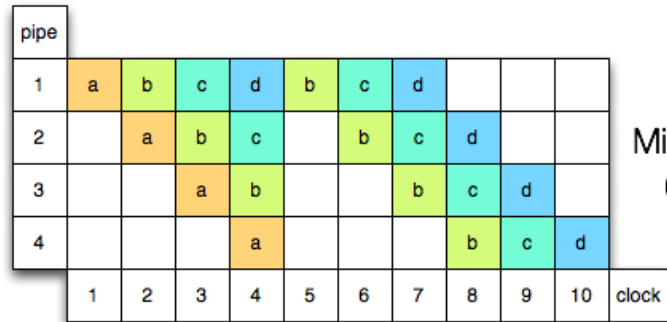
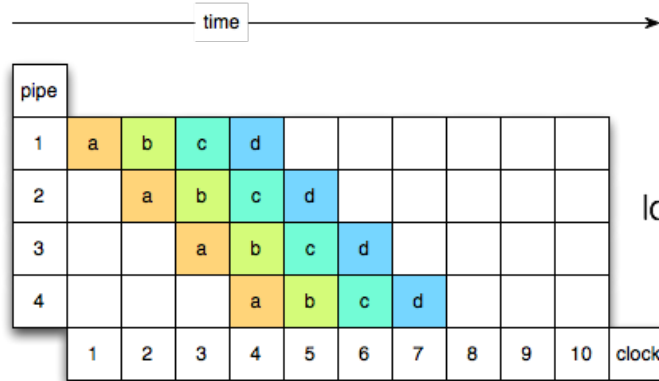
- Nested Data Structures
- Maximize CPU throughput
 - Pipelining
 - SIMD
 - cache locality
- Scatter/gather I/O



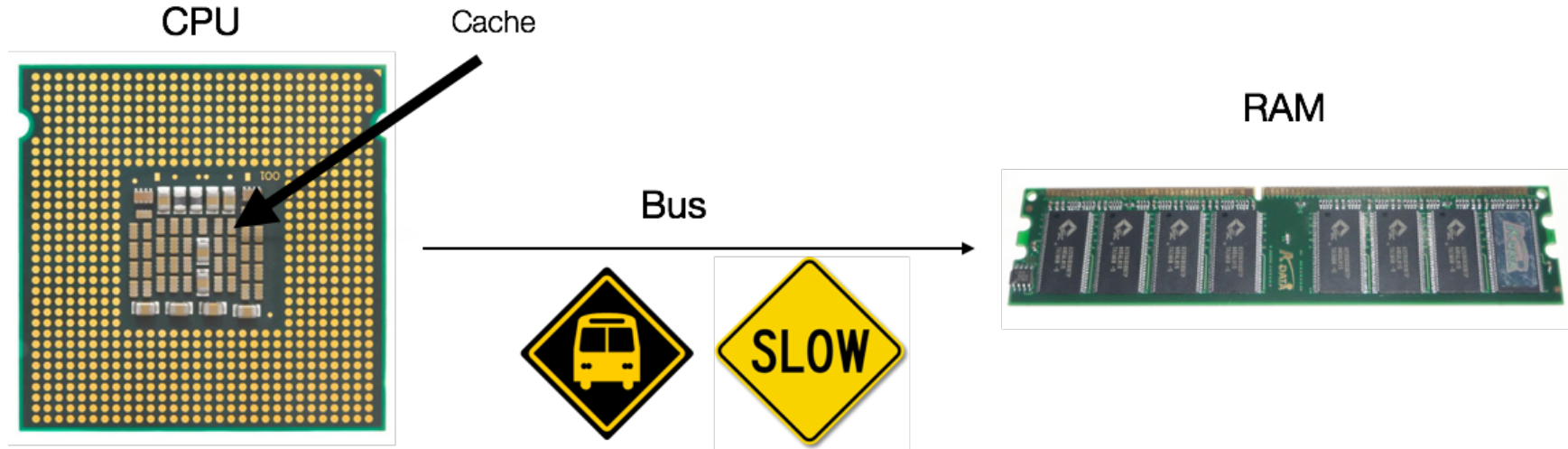
CPU pipeline



pipeline



Minimize CPU cache misses



a cache miss costs 10 to 100s cycles depending on the level



Focus on CPU Efficiency

- Cache Locality
- Super-scalar & vectorized operation
- Minimal Structure Overhead
- Constant value access
 - With minimal structure overhead
- Operate directly on columnar compressed data

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

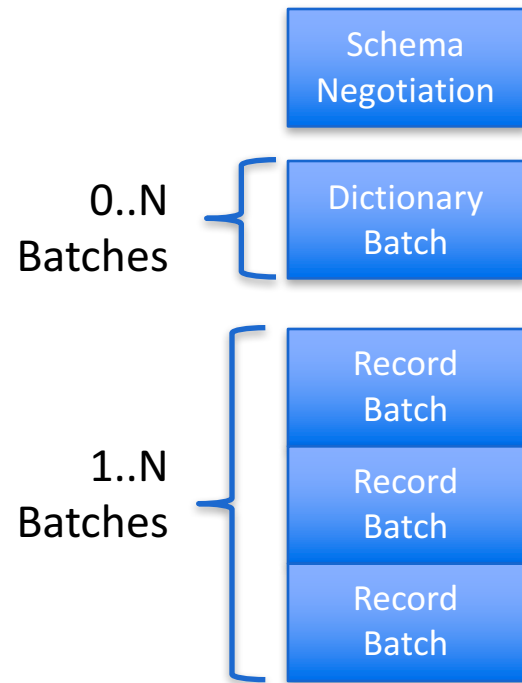


Arrow Messages, RPC & IPC



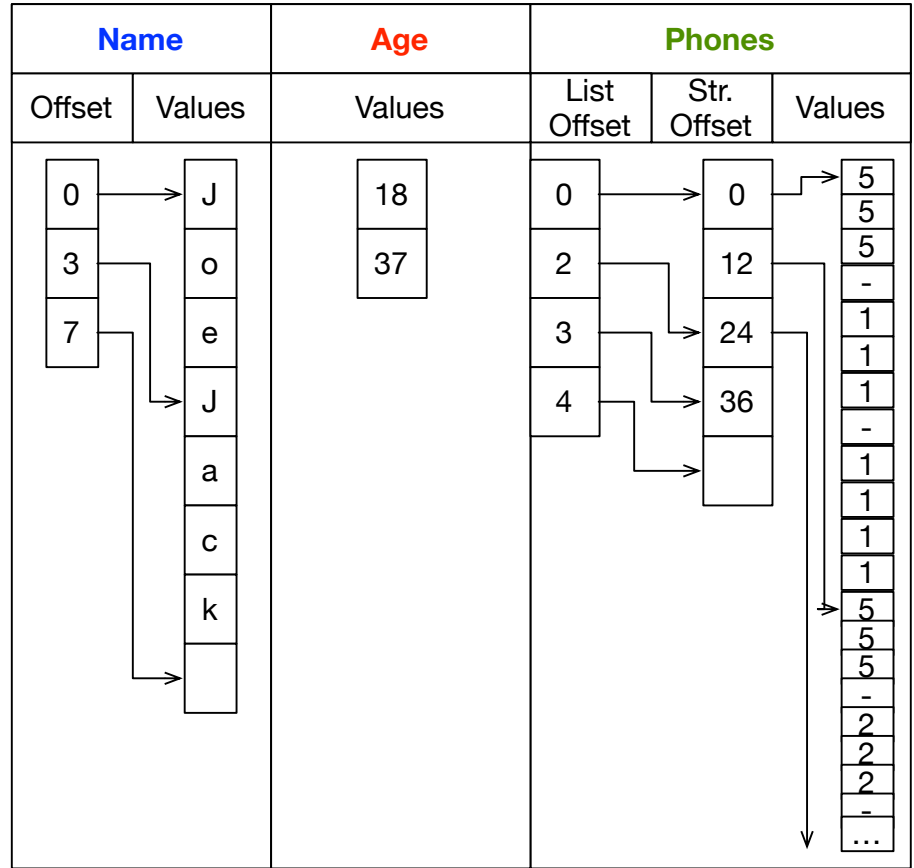
Common Message Pattern

- Schema Negotiation
 - Logical Description of structure
 - Identification of dictionary encoded Nodes
- Dictionary Batch
 - Dictionary ID, Values
- Record Batch
 - Batches of records up to 64K
 - Leaf nodes up to 2B values

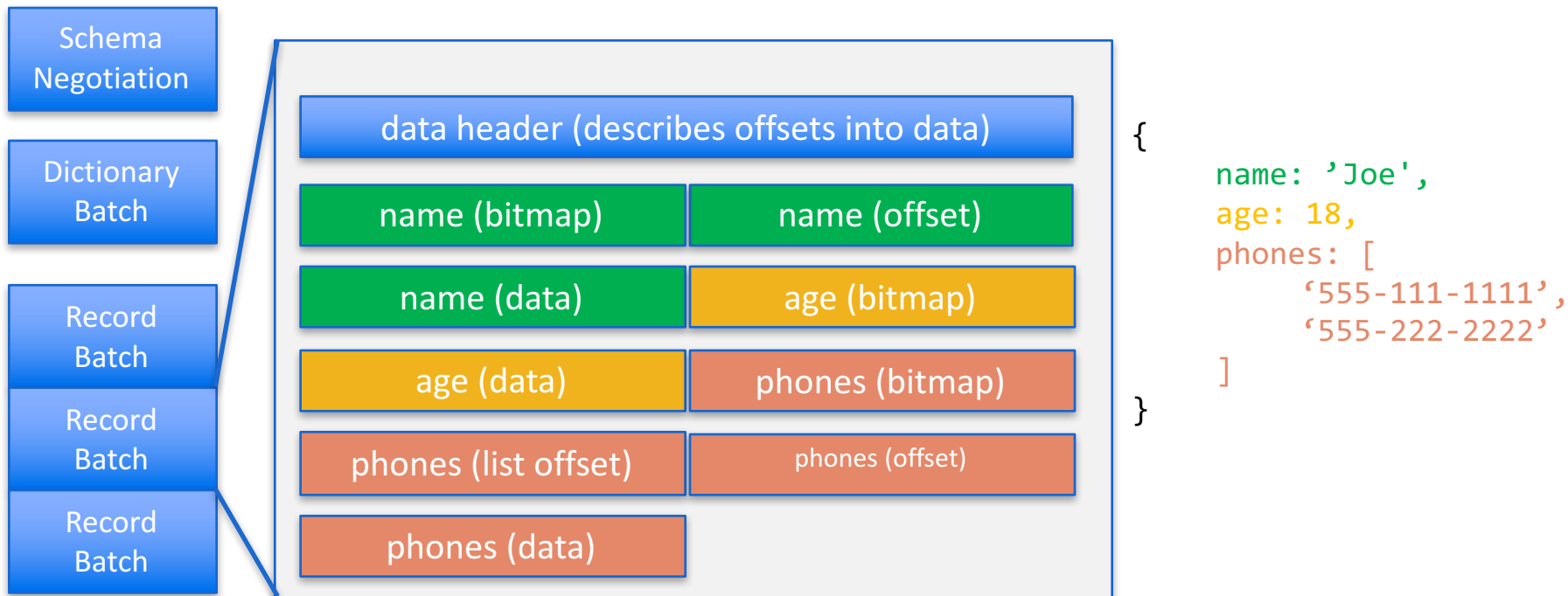


Columnar data

```
persons = [{  
  name: 'Joe',  
  age: 18,  
  phones: [  
    '555-111-1111',  
    '555-222-2222'  
  ]  
}, {  
  name: 'Jack',  
  age: 37,  
  phones: [ '555-333-3333' ]  
}]
```



Record Batch Construction



Each box (vector) is contiguous memory
The entire record batch is contiguous on wire



Moving Data Between Systems

RPC

- Avoid Serialization & Deserialization
- Layer TBD: Focused on supporting vectored io
 - Scatter/gather reads/writes against socket

IPC

- Alpha implementation using memory mapped files
 - Moving data between Python and Drill
- Working on shared allocation approach
 - Shared reference counting and well-defined ownership semantics



Java: Memory Management

- Chunk-based managed allocator
 - Built on top of Netty's JEMalloc implementation
- Create a tree of allocators
 - Limit and transfer semantics across allocators
 - Leak detection and location accounting
- Wrap native memory from other applications



Language Bindings

Parquet

- Target Languages
 - Java
 - CPP
 - Python & Pandas
- Engines integration:
 - Many!

Arrow

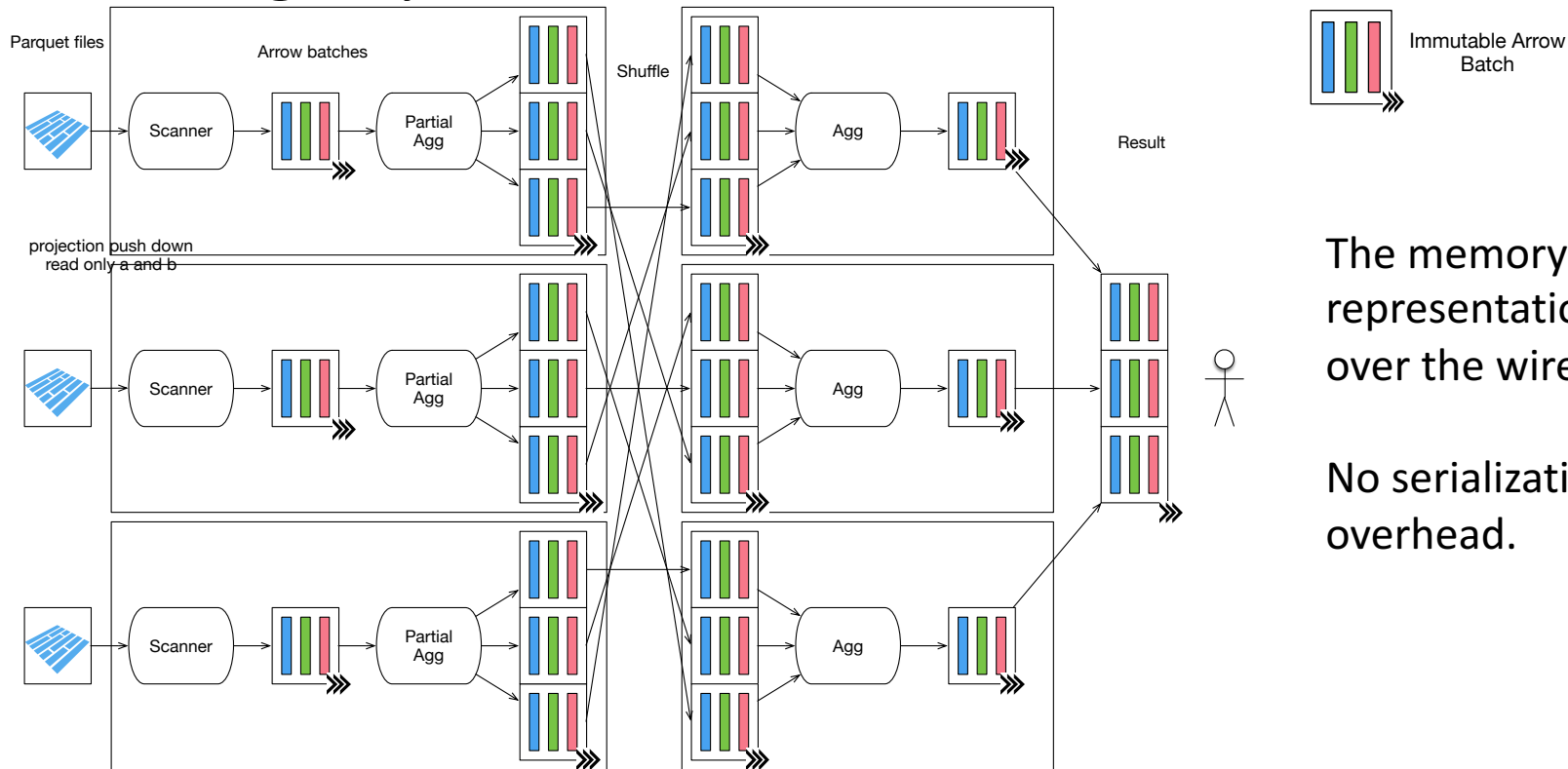
- Target Languages
 - Java
 - CPP, Python
 - R (underway)
- Engines integration:
 - Drill
 - Pandas, R
 - Spark (underway)



Execution examples:



RPC: Single system execution



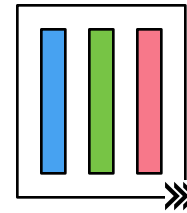
SELECT SUM(a) FROM t GROUP BY b

The memory representation is sent over the wire.

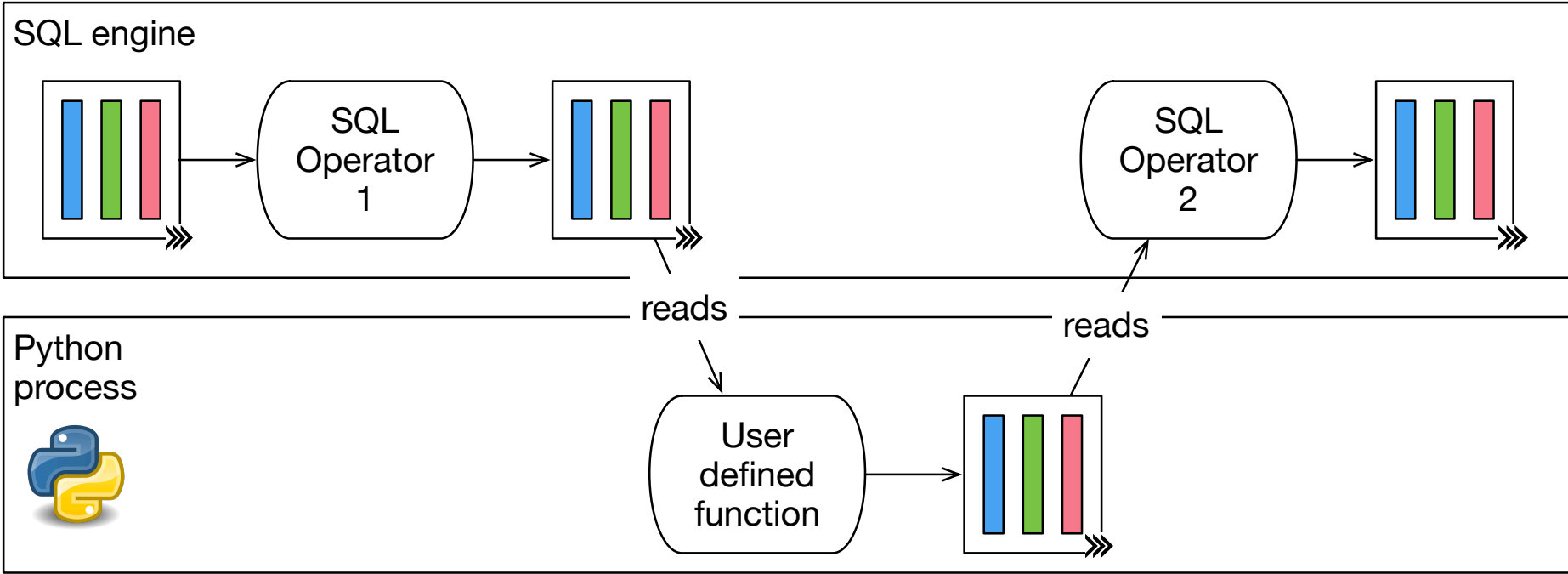
No serialization overhead.



Multi-system IPC



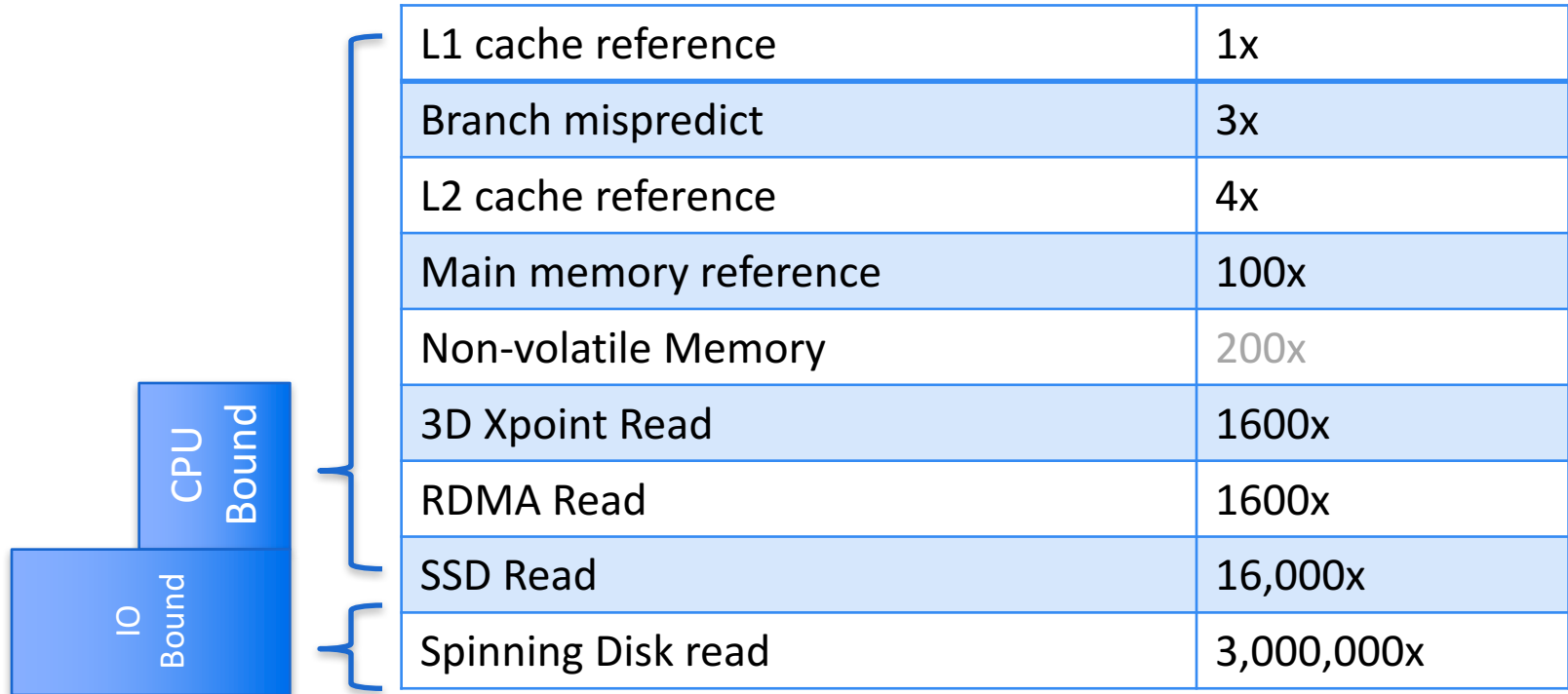
Immutable Arrow Batch



Summary and Future



Where is the bottleneck?

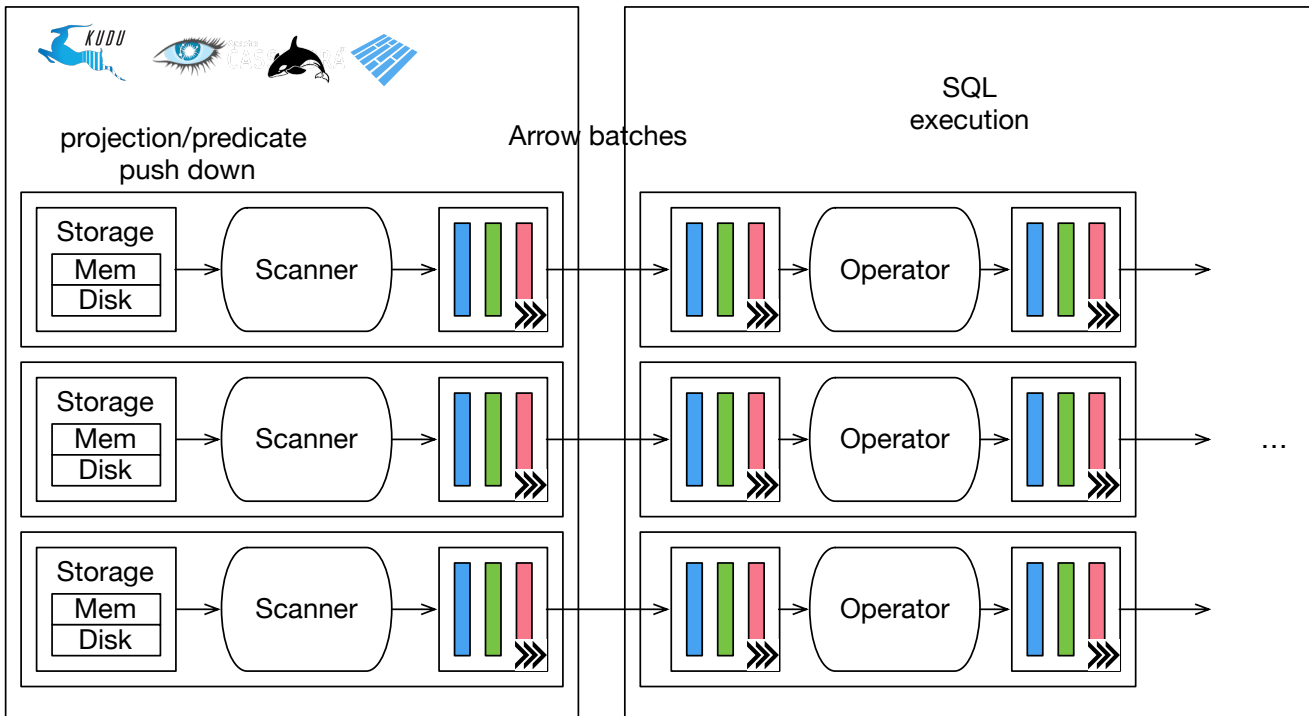


Sources

- https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html
- <http://www.anandtech.com/show/9470/intel-and-micron-announce-3d-xpoint-nonvolatile-memory-technology-1000x-higher-performance-endurance-than-nand>



RPC: arrow based storage interchange

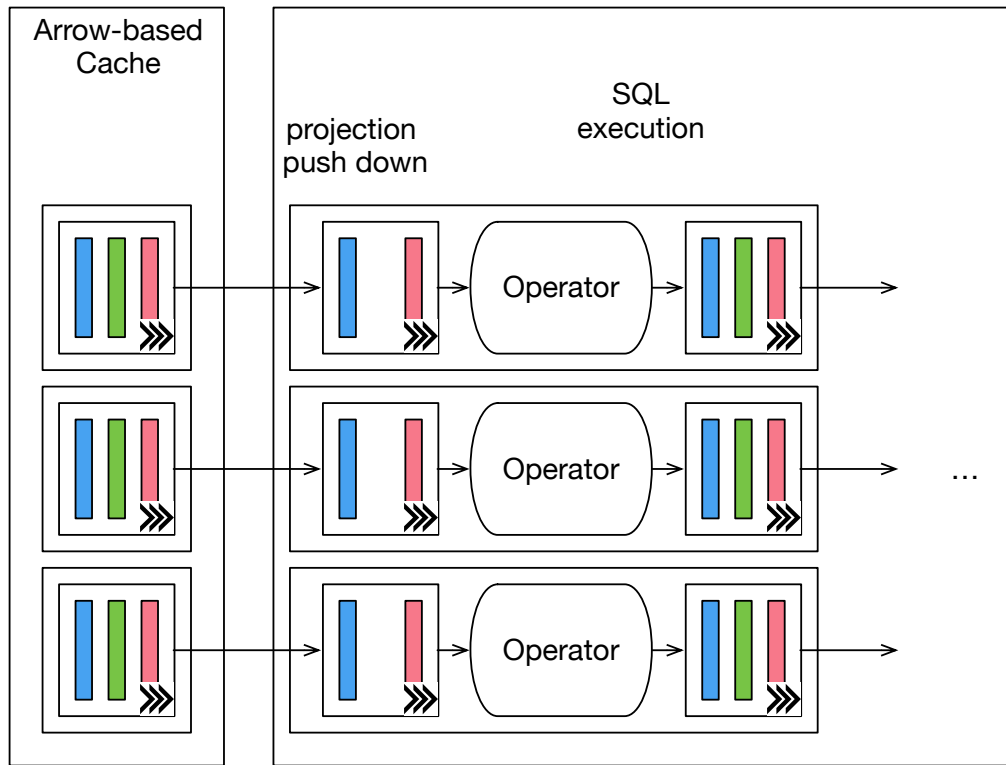


The memory representation is sent over the wire.

No serialization overhead.



RPC: arrow based cache

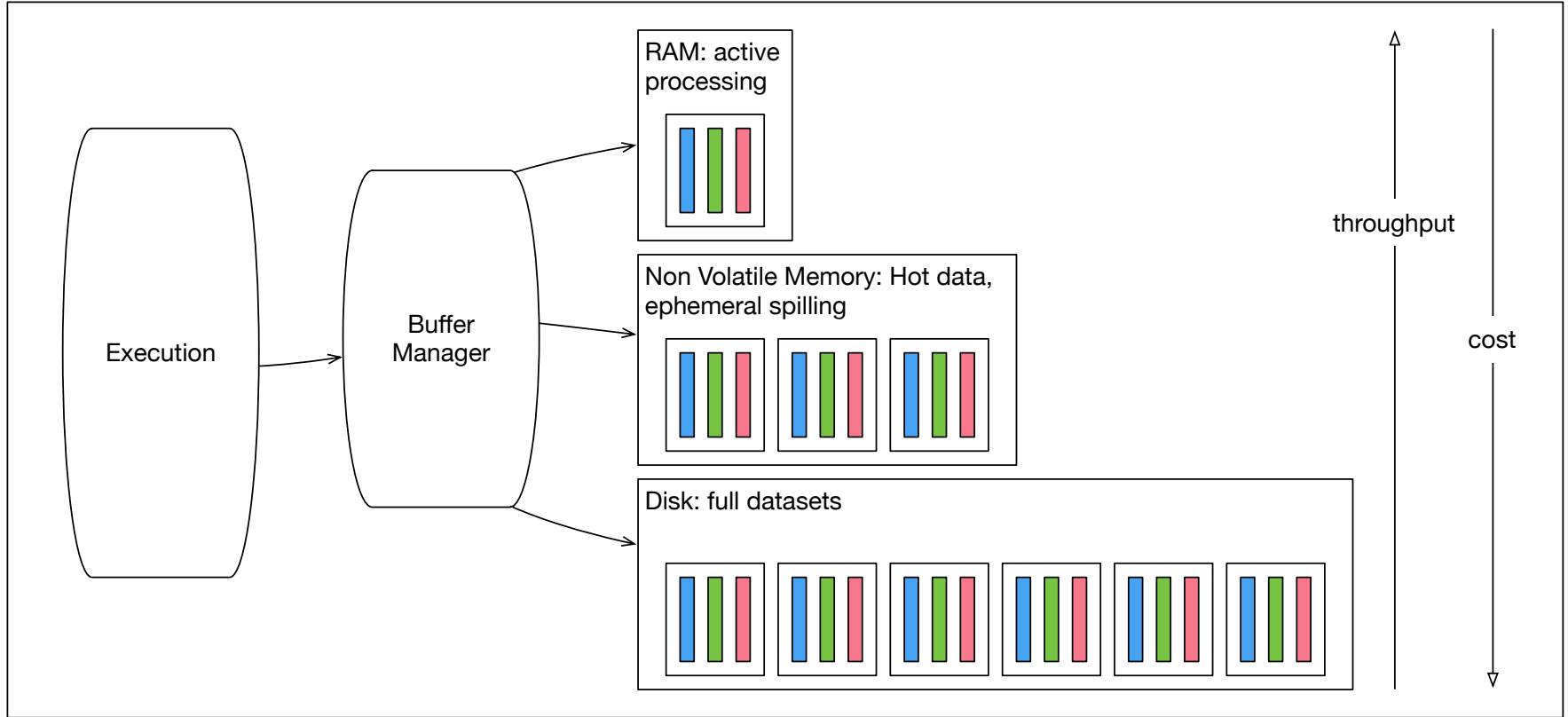


The memory representation is sent over the wire.

No serialization overhead.



Storage tiering with Non Volatile Memory



What's Next

- Parquet – Arrow Nested support for Python & C++
- Arrow IPC Implementation
- Arrow RPC & HTTP/2
- Kudu – Arrow integration
- Apache {Spark, Drill} to Arrow Integration
 - Faster UDFs, Storage interfaces
- Support for integration with Intel's Persistent Memory library via Apache Mnemonic



Get Involved

- Join the community
 - dev@{arrow,parquet}.apache.org
 - Slack:
 - <https://apachearrowslackin.herokuapp.com/>
 - http://{arrow,parquet}.apache.org
 - Follow @Apache{Parquet,Arrow}

