Towards Faster Columnar Data Transport Using RDMA

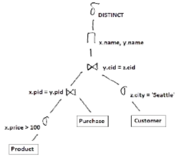
Jayjeet Chakraborty

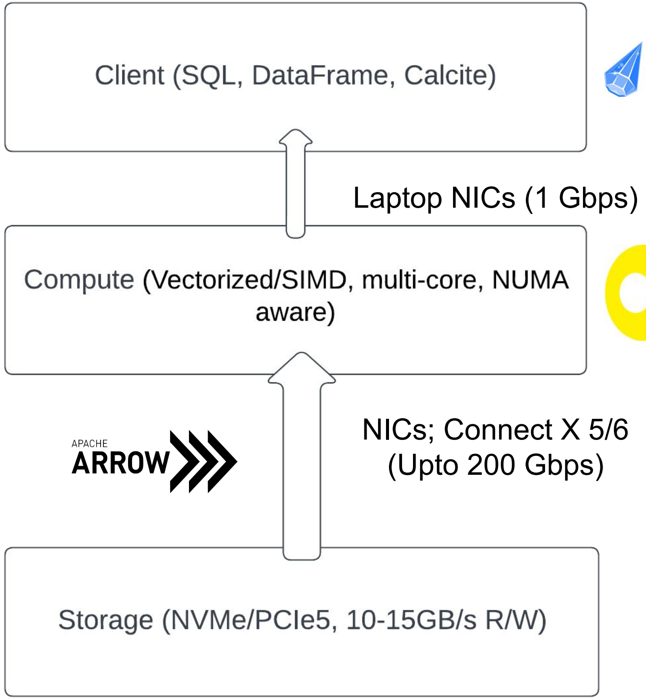
UC Santa Cruz

Modern Datacenter Hardware

- **Fast memory devices**
  - NVMe
  - PCIe5, DDR5, CXL
- **Fast networking infrastructure**
  - ConnectX-5/6 NICs
  - Upto 200 Gbps bandwidth
- **Fat CPUs**
  - Intel Xeon
  - Intel Sapphire Rapids

Data Processing Architecture using CS





Client (SQL, DataFrame, Calcite)

Laptop NICs (1 Gbps)

Compute (Vectorized/SIMD, multi-core, NUMA aware)

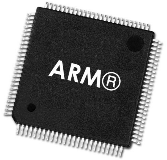
NICs; Connect X 5/6 (Upto 200 Gbps)

Storage (NVMe/PCIe5, 10-15GB/s R/W)

APACHE calcite™

DuckDB

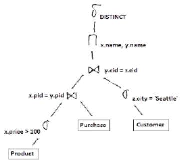
APACHE ARROW

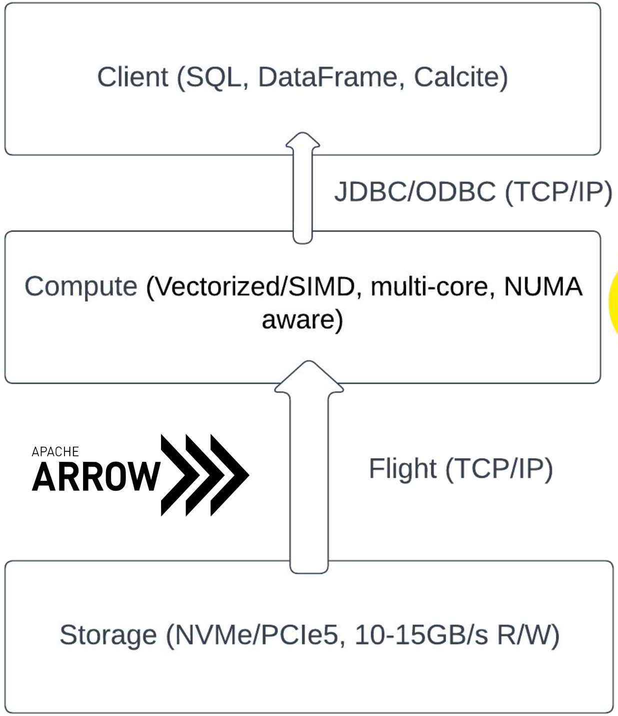


ceph

DuckDB

Data Processing Architecture using CS





Client (SQL, DataFrame, Calcite)

JDBC/ODBC (TCP/IP)

Compute (Vectorized/SIMD, multi-core, NUMA aware)

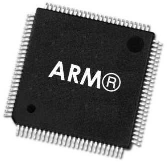
Flight (TCP/IP)

Storage (NVMe/PCIe5, 10-15GB/s R/W)

APACHE ARROW

APACHE calcite™

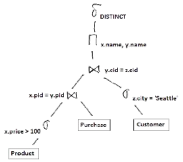
DuckDB

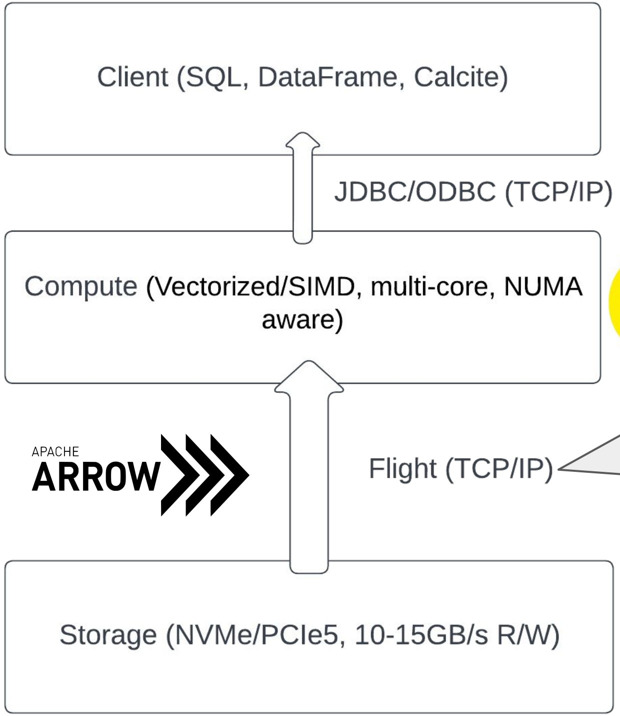


ceph

DuckDB

Data Processing Architecture using CS





Client (SQL, DataFrame, Calcite)

JDBC/ODBC (TCP/IP)

Compute (Vectorized/SIMD, multi-core, NUMA aware)

APACHE ARROW

Flight (TCP/IP)

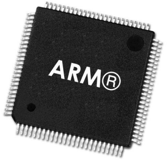
Storage (NVMe/PCIe5, 10-15GB/s R/W)

APACHE calcite™

Inherently TCP/IP-based
Does not leverage modern network protocols
Serialization

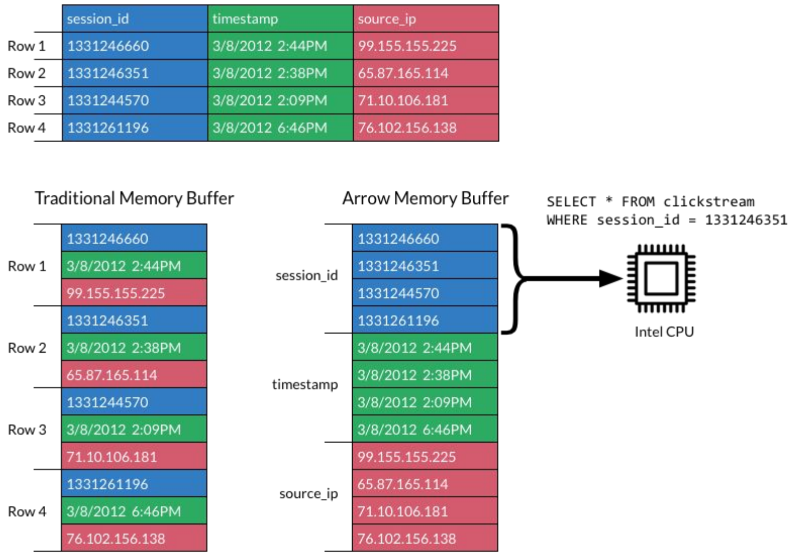
ceph

DuckDB



What is Serialization ?

The process of converting 2D tables/record batches into network transferable 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

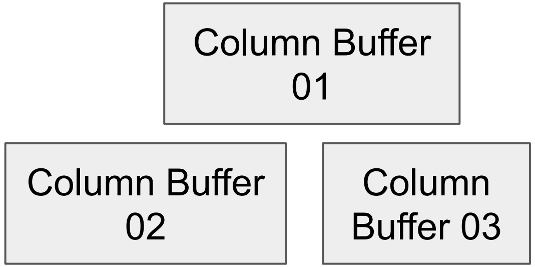
Traditional Memory Buffer

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

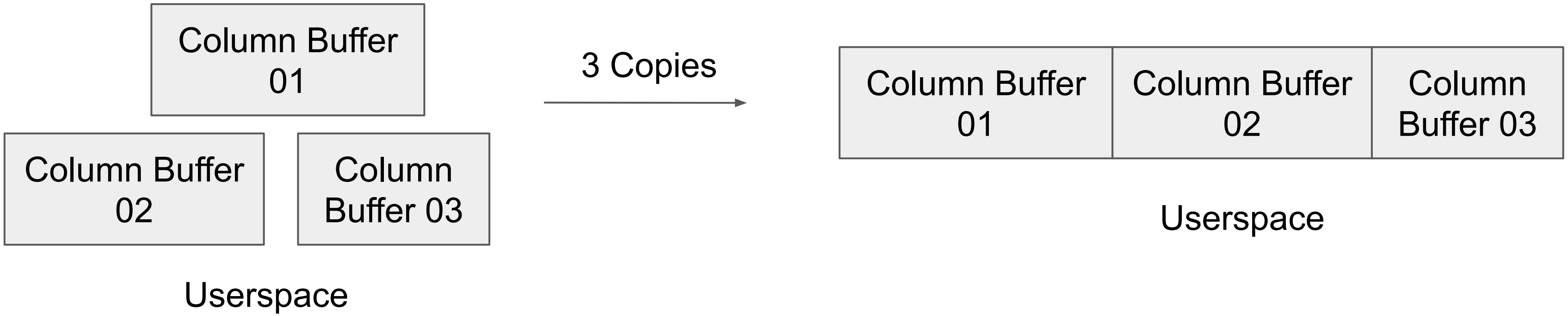
session_id	1331246660
	1331246351
	1331244570
	1331261196
timestamp	3/8/2012 2:44PM
	3/8/2012 2:38PM
	3/8/2012 2:09PM
	3/8/2012 6:46PM
source_ip	99.155.155.225
	65.87.165.114
	71.10.106.181
	76.102.156.138

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



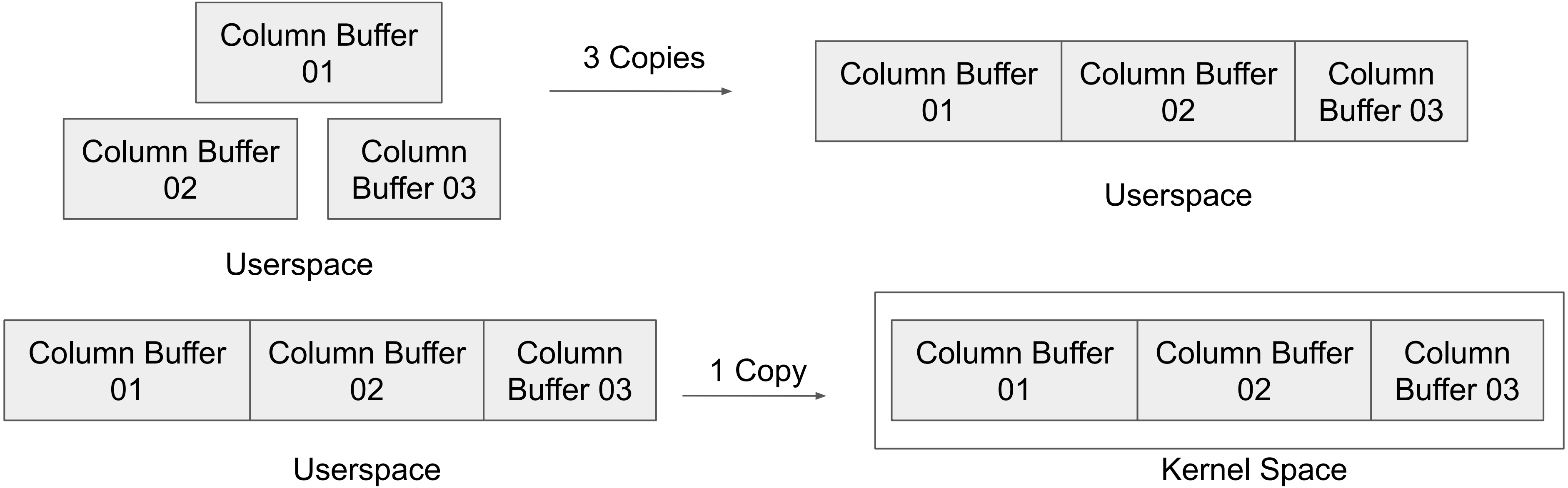
What is Serialization ?

Copy the individual buffers holding tabular data into a single-contiguous buffer as required by TCP/IP



What is Serialization ?

Copy the individual buffers holding tabular data into a single-contiguous buffer as required by TCP/IP



Column Buffer 01

Column Buffer 02

Column Buffer 03

Userspace

3 Copies

Column Buffer 01 | Column Buffer 02 | Column Buffer 03

Userspace

Column Buffer 01 | Column Buffer 02 | Column Buffer 03

Userspace

1 Copy

Column Buffer 01 | Column Buffer 02 | Column Buffer 03

Kernel Space

Why is Serialization bad ?

- Unwanted memory copies
- Wastage of CPU cycles
- Added overhead for Computational storage

Why is Serialization bad ?

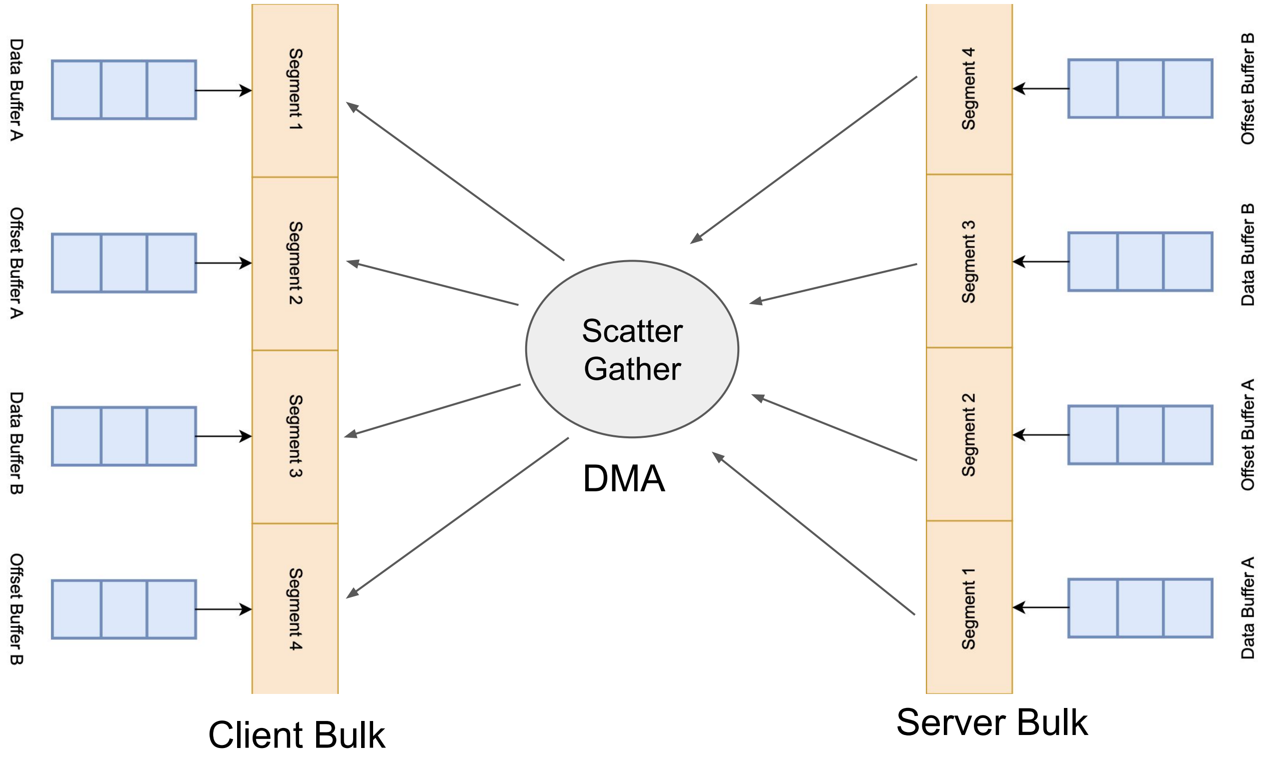
- Unwanted memory copies
- Wastage of CPU cycles
- Added overhead for Computational Storage

How much can we eliminate the serialization overhead ?

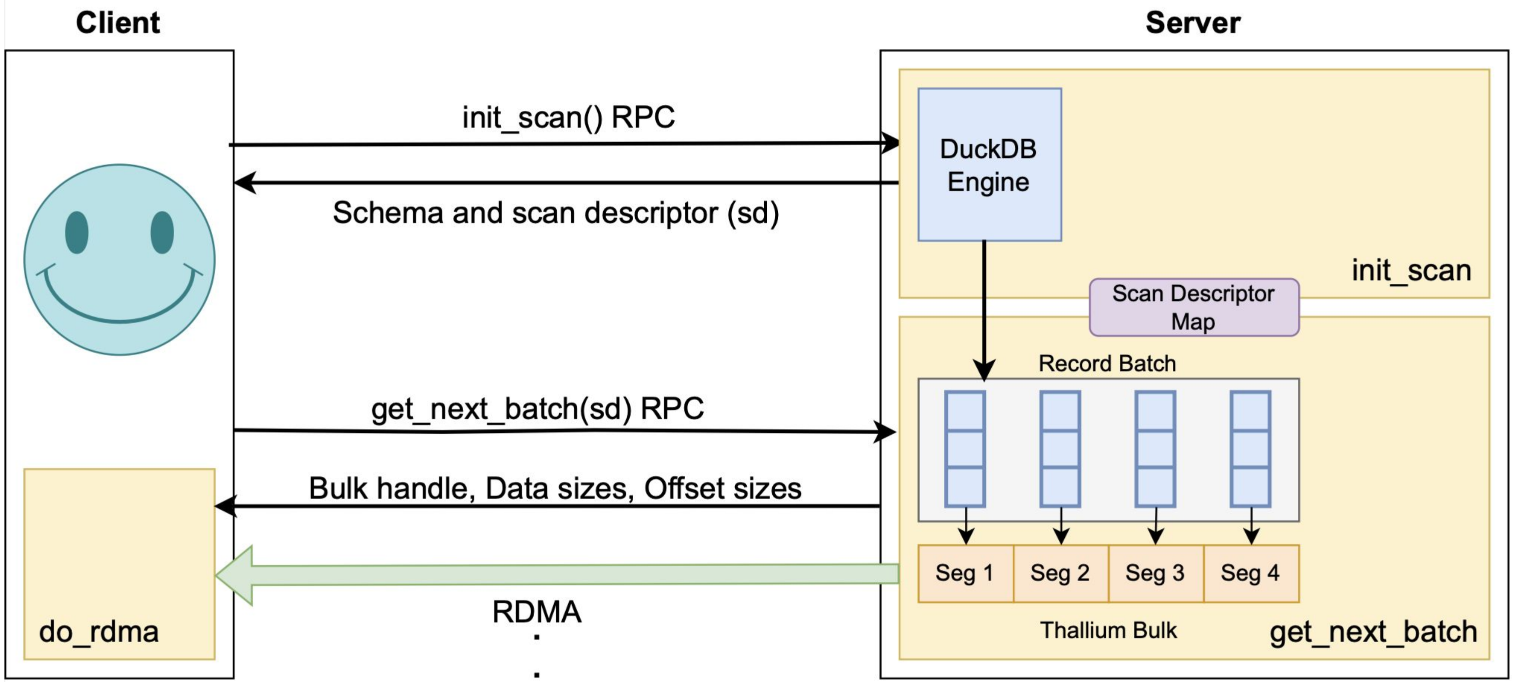
Possible Solution

- Eliminate the multiple rounds of `memcpy`
- Use user-space networking libraries
- **Leverage HPC communication frameworks that leverage faster networking protocols**
    - Mochi Thallium (Argonne National Labs)
        - Supports Infiniband; VPI-enabled ConnectX cards has both Ethernet and Infiniband modes
        - Uses `user-space` RDMA libraries; libfabric and libibverbs

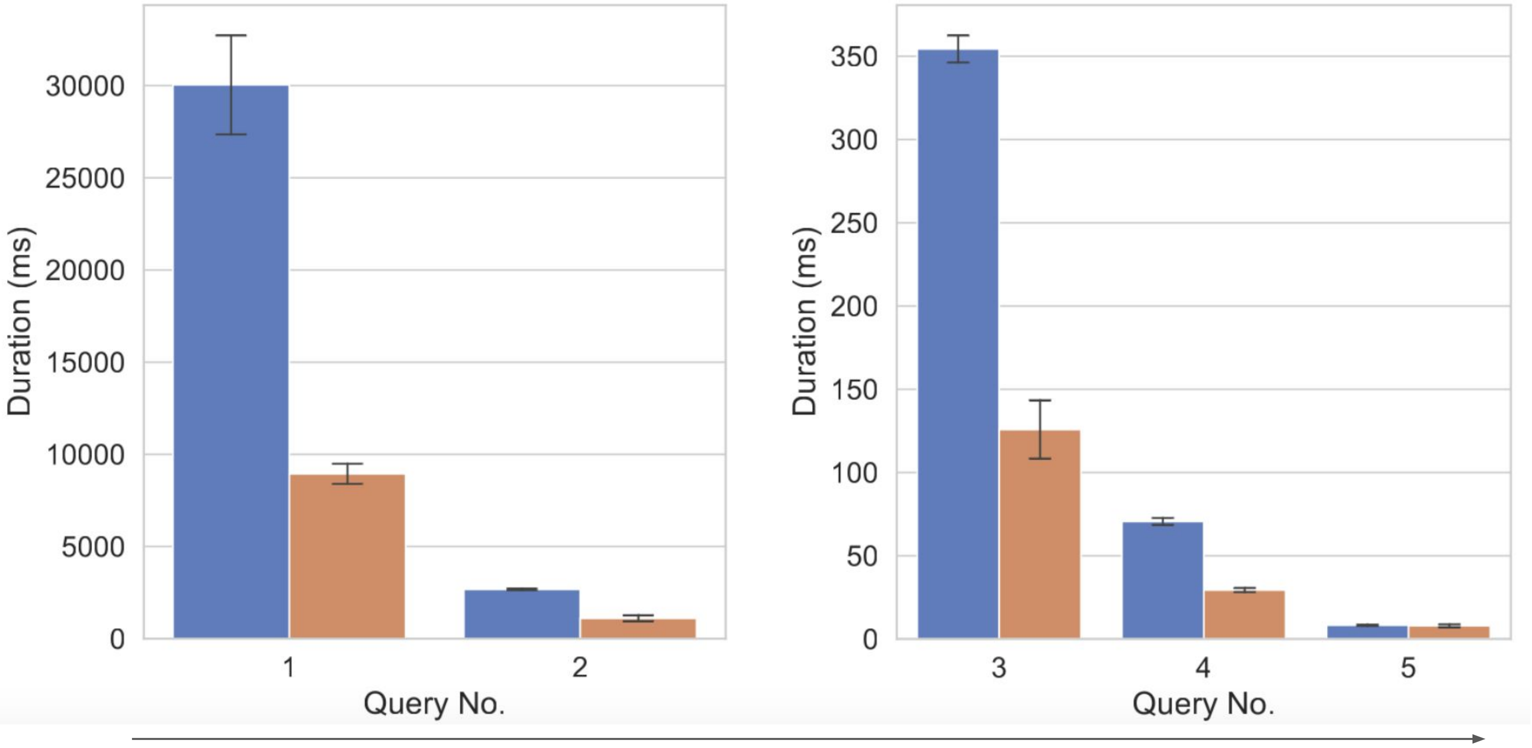
Mochi Thallium



Protocol Design

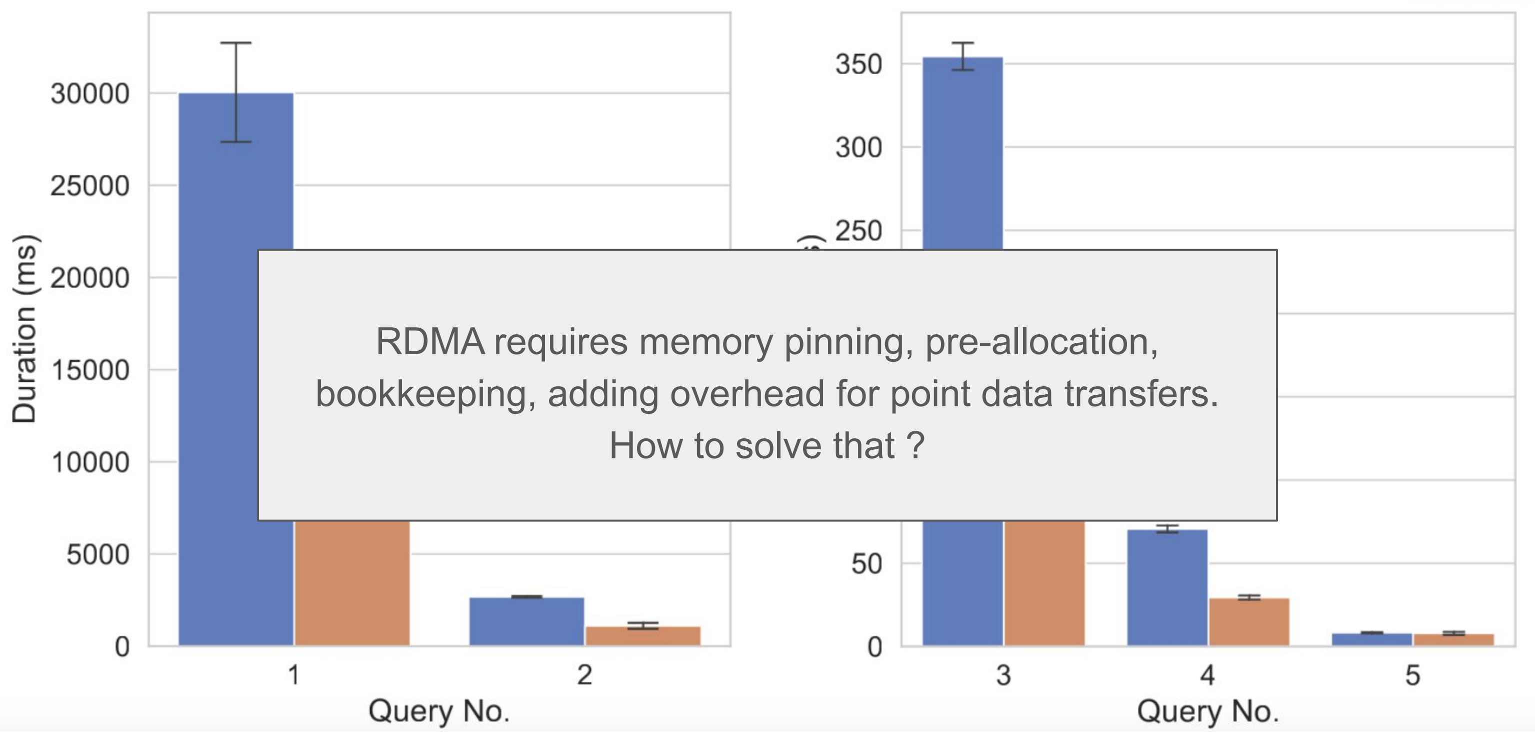


Initial Evaluations (with DuckDB engine)



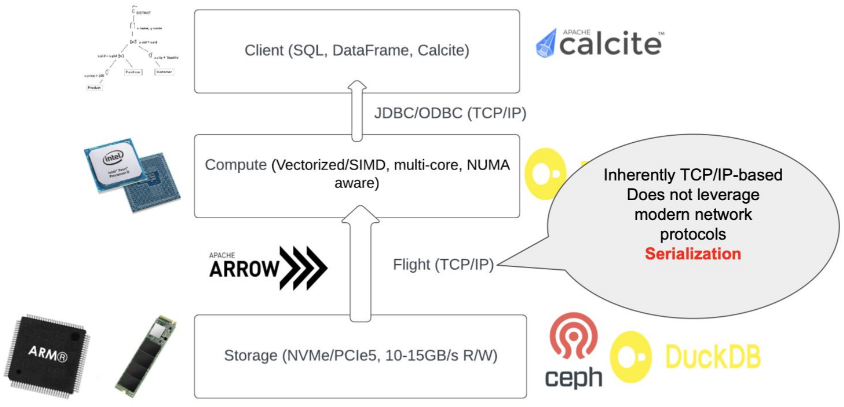
Higher Selectivity, Smaller result size

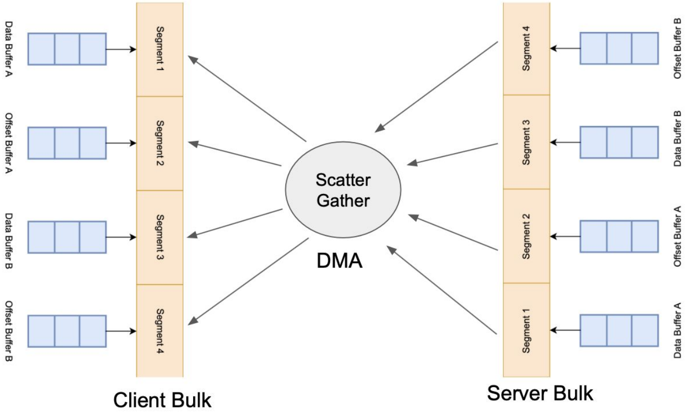
Initial Evaluations (with DuckDB engine)



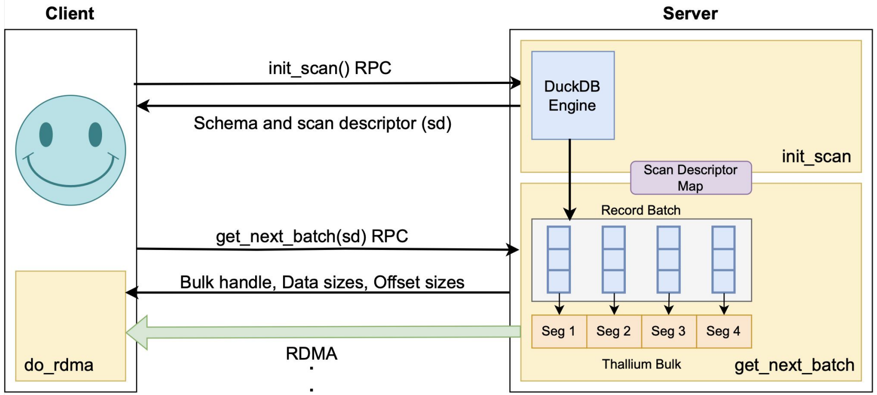
RDMA requires memory pinning, pre-allocation, bookkeeping, adding overhead for point data transfers. How to solve that ?

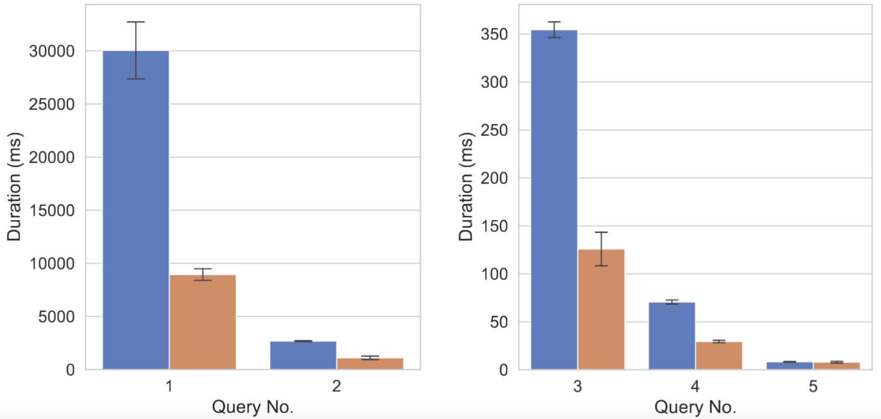
Higher Selectivity, Smaller result size


Client (SQL, DataFrame, Calcite)
APACHE calcite
JDBC/ODBC (TCP/IP)
Compute (Vectorized/SIMD, multi-core, NUMA aware)
Inherently TCP/IP-based
Does not leverage modern network protocols
Serialization
APACHE ARROW
Flight (TCP/IP)
Storage (NVMe/PCIe5, 10-15GB/s R/W)
ARM®
ceph
DuckDB



Thank You ! (jayjeetc@ucsc.edu)


Client
Server
init_scan() RPC
Schema and scan descriptor (sd)
DuckDB Engine
init_scan
Scan Descriptor Map
Record Batch
get_next_batch(sd) RPC
Bulk handle, Data sizes, Offset sizes
Seg 1
Seg 2
Seg 3
Seg 4
RDMA
do_rdma
Thallium Bulk
get_next_batch


Duration (ms)
Query No.
Duration (ms)
Query No.