ETH zürich

Farview

Disaggregated Memory with Operator Off-loading for Database Engines

Dario Korolija[*], Dimitrios Koutsoukos[*], Kimberly Keeton[±], Konstantin Taranov[*], Dejan Milojičić[¥], Gustavo Alonso[*]

[*] *Systems Group, Dept. of Computer Science, ETH Zurich*
[¥] *Hewlett Packard Enterprise*
[±] *Hewlett Packard Enterprise (now at Google)*

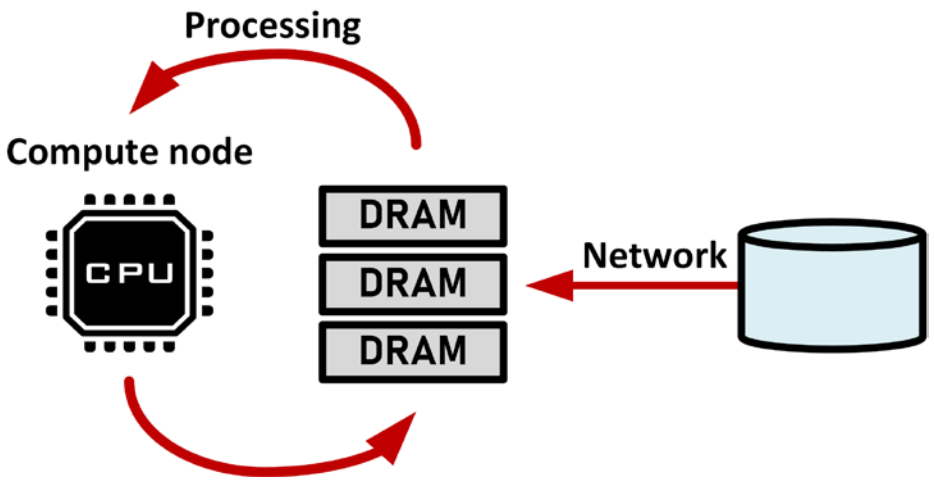
Systems@ETH zürich DINFK

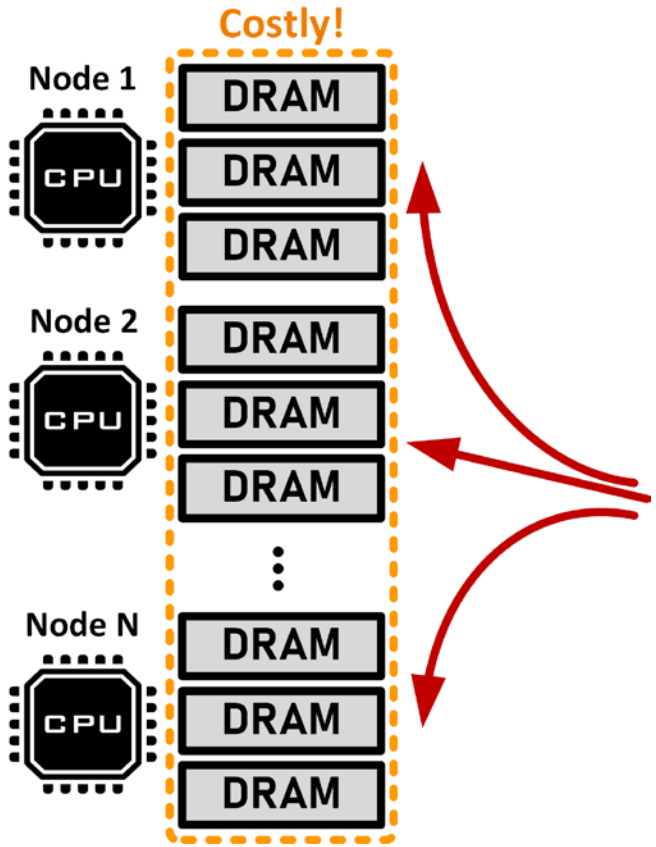
Farview

Motivation

- The I/O overheads one of the main factors in the overall performance
- More and more data kept in local **DRAM** of compute nodes

1. **Excessive data movement**
2. **Memory capacity limitation**





Farview

Motivation

Disaggregation of compute, memory and storage

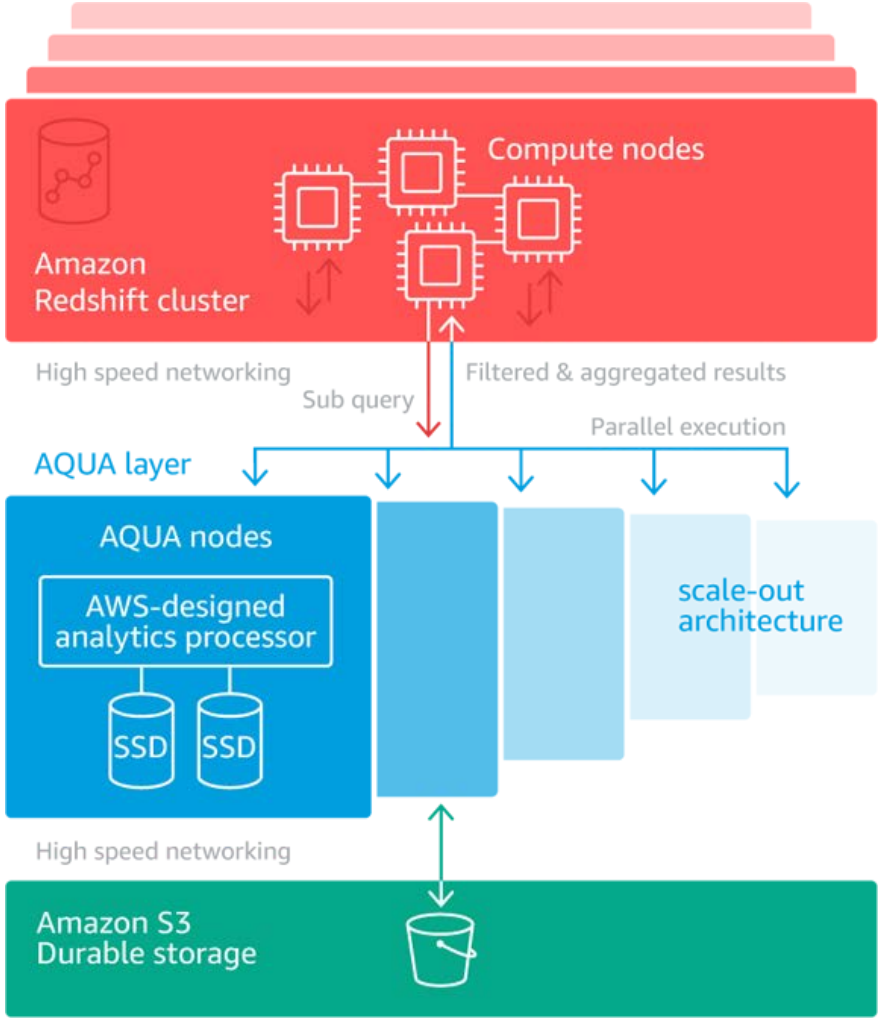
Farview

Motivation

- FPGAs as smart accelerator for disaggregated resources

- Amazon **AQUA**
  https://aws.amazon.com/blogs/aws/new-aqua-advanced-query-accelerator-for-amazon-redshift/

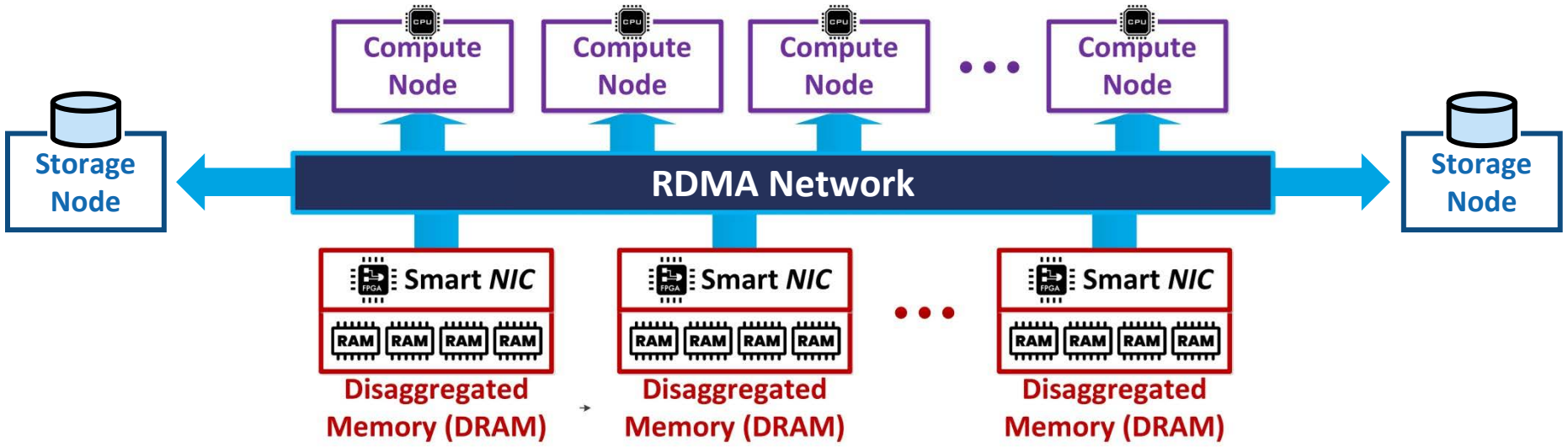
- Offload of analytic computation to FPGA
- Pushing computation closer to data
- Reduces data movement
- Reduces CPU and network overheads



Farview

Introduction

- **FPGA-based smart NIC** making **DRAM** available as a pool of network attached memory, provisioned on demand, accessible over high performance **RDMA**.
- Capability to perform line-rate data processing with minimal overheads
- **Farview acts as a disaggregated buffer cache with operator pushdown capabilities**



Farview

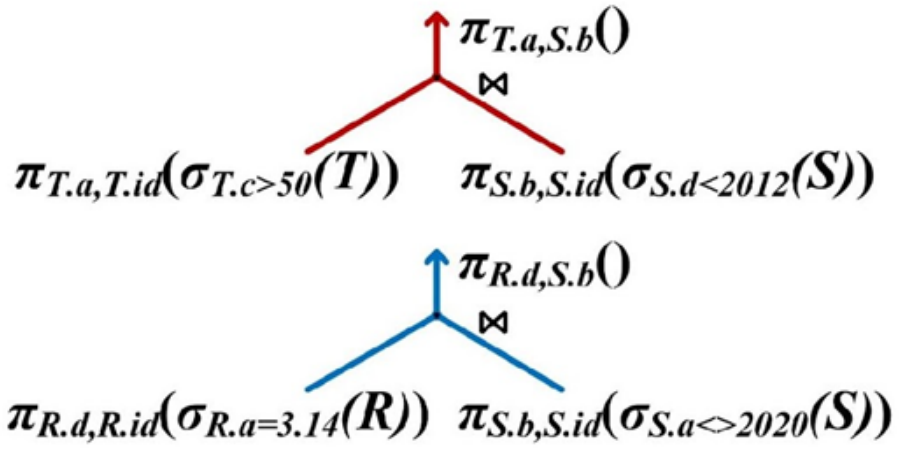
Overview

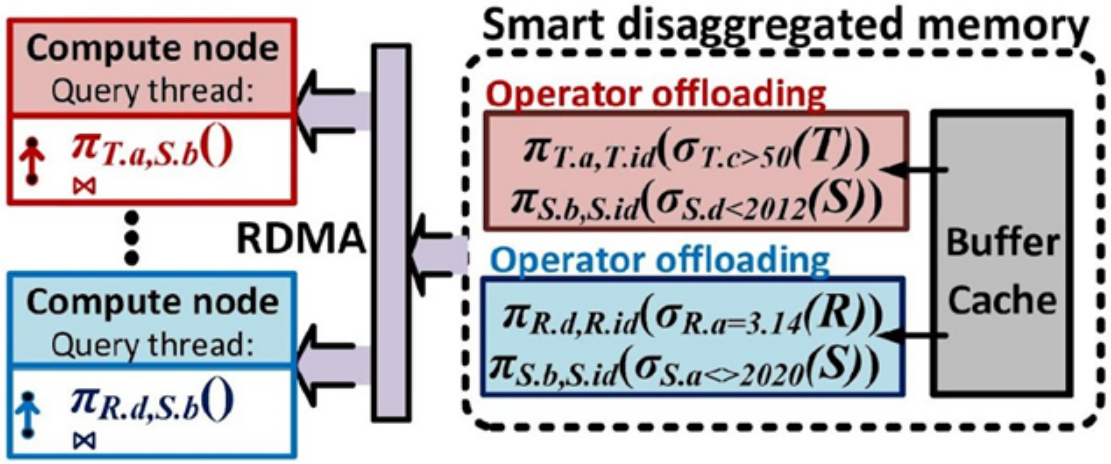
- **Farview** positioned to address the issues of inefficient data movement and memory capacity limitations
- Consider the following two queries:

```
SELECT T.a, S.b
FROM T, S
WHERE T.id = S.id
AND T.c > 50 AND S.d < 2012;
```

```
SELECT R.d, S.b
FROM R, S
WHERE R.id = S.id
AND R.a = 3.14 AND S.a <> 2020;
```

- **Farview** centralizes the buffer cache in disaggregated memory and pushes down operators

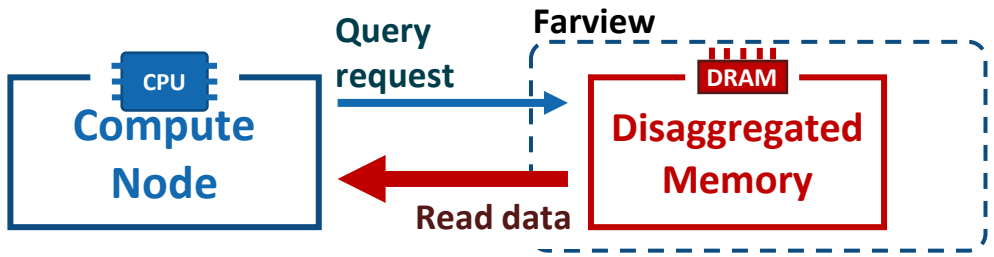




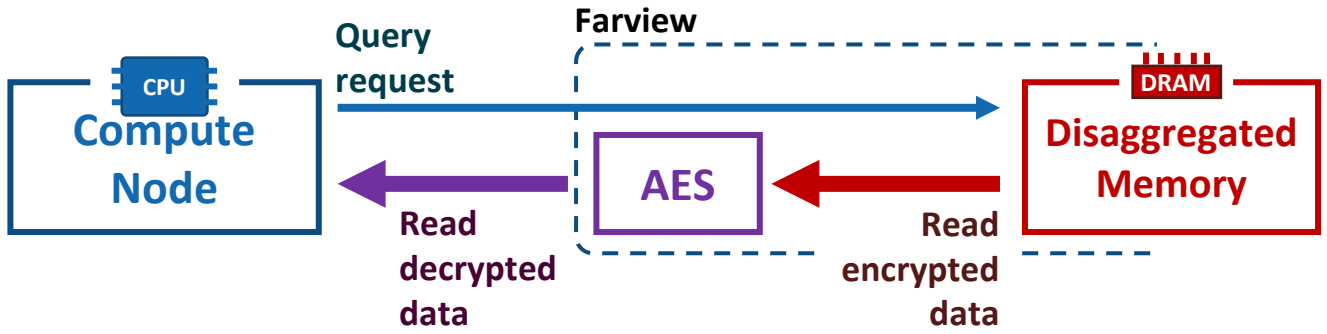
Farview

Example

1. No processing in **Farview**, simple **READ** operation:

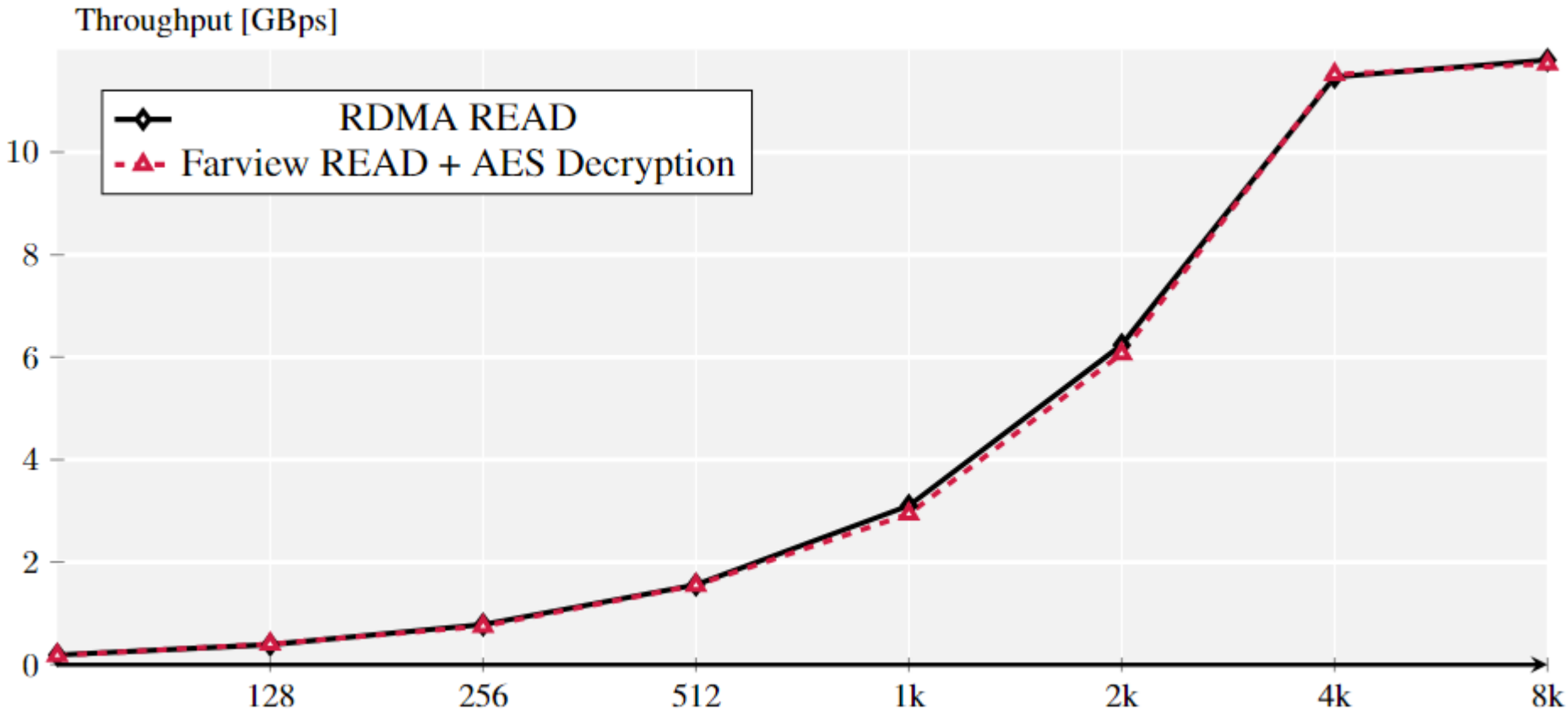


2. AES decryption on the same data as it is being read:



Farview

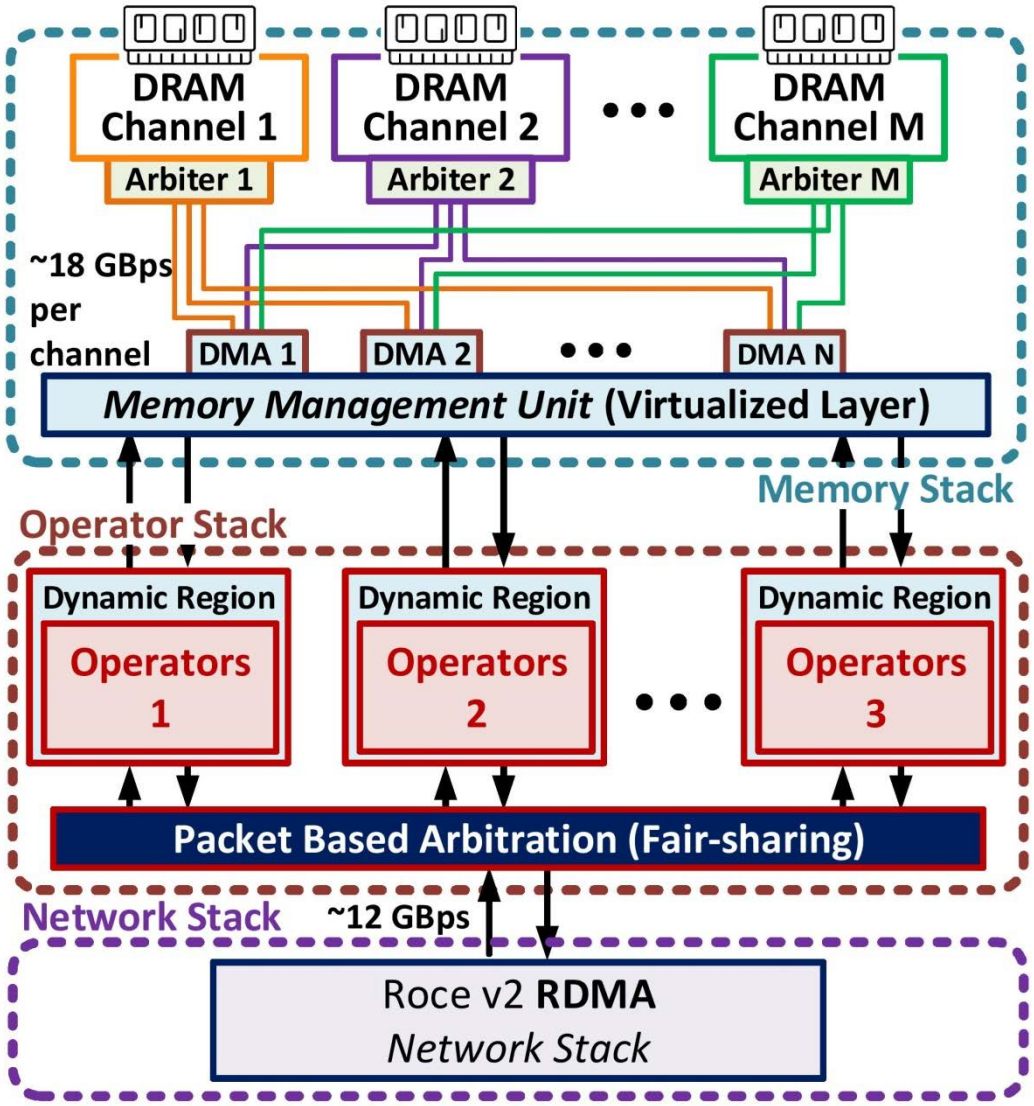
Example



Farview

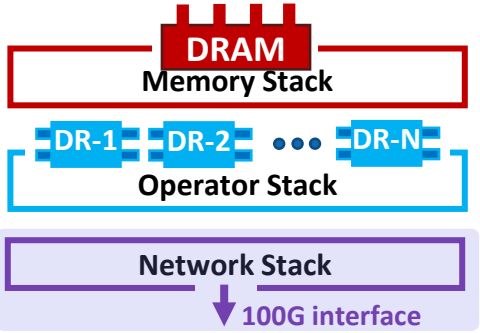
System Architecture

- Several components needed:
  - DRAM (or HBM)
  - Memory controllers
  - Memory management unit
  - Network stack
  - Mechanism to support concurrent accesses to the memory
  - Stream processing capacity
  - Mechanism to swap the operators
- Three distinct modules
  - Operator stack
  - Memory stack
  - Network stack

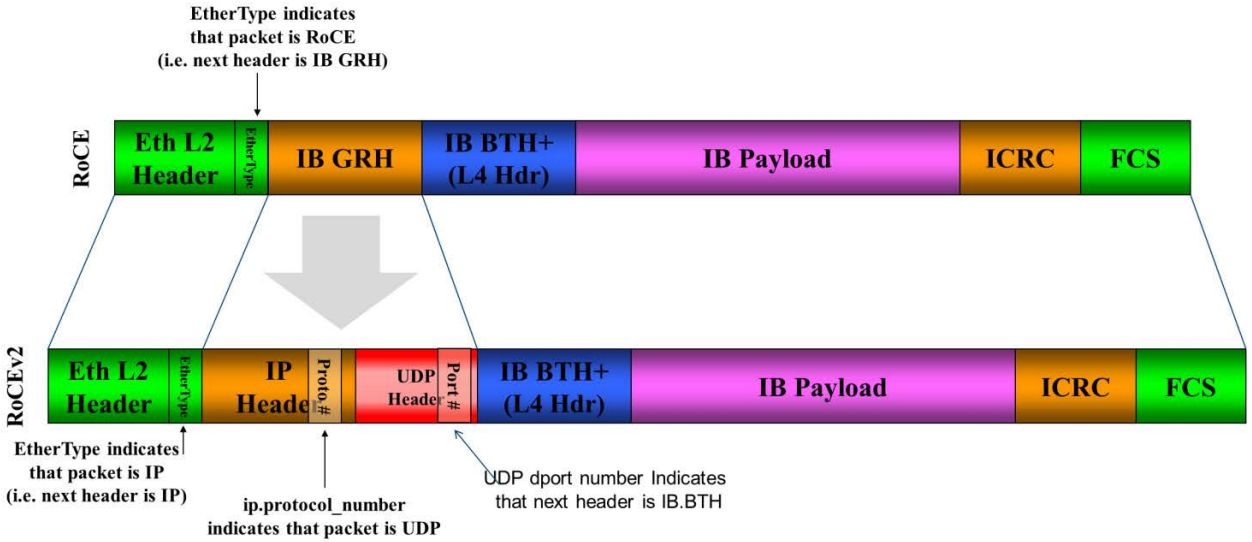


Farview

Network stack (RDMA)



- Open source RDMA stack (UC, RC)[1]
- RDMA over Converged Ethernet (RoCE v2)
- Implemented on top of UDP/IPv4/IPv6 (far lower overhead than iWARP)
- InfiniBand (IB) transport packets over Ethernet (READ, WRITE, SEND)



https://docs.nvidia.com/networking/display/WINOFv55053000/RoCEv2

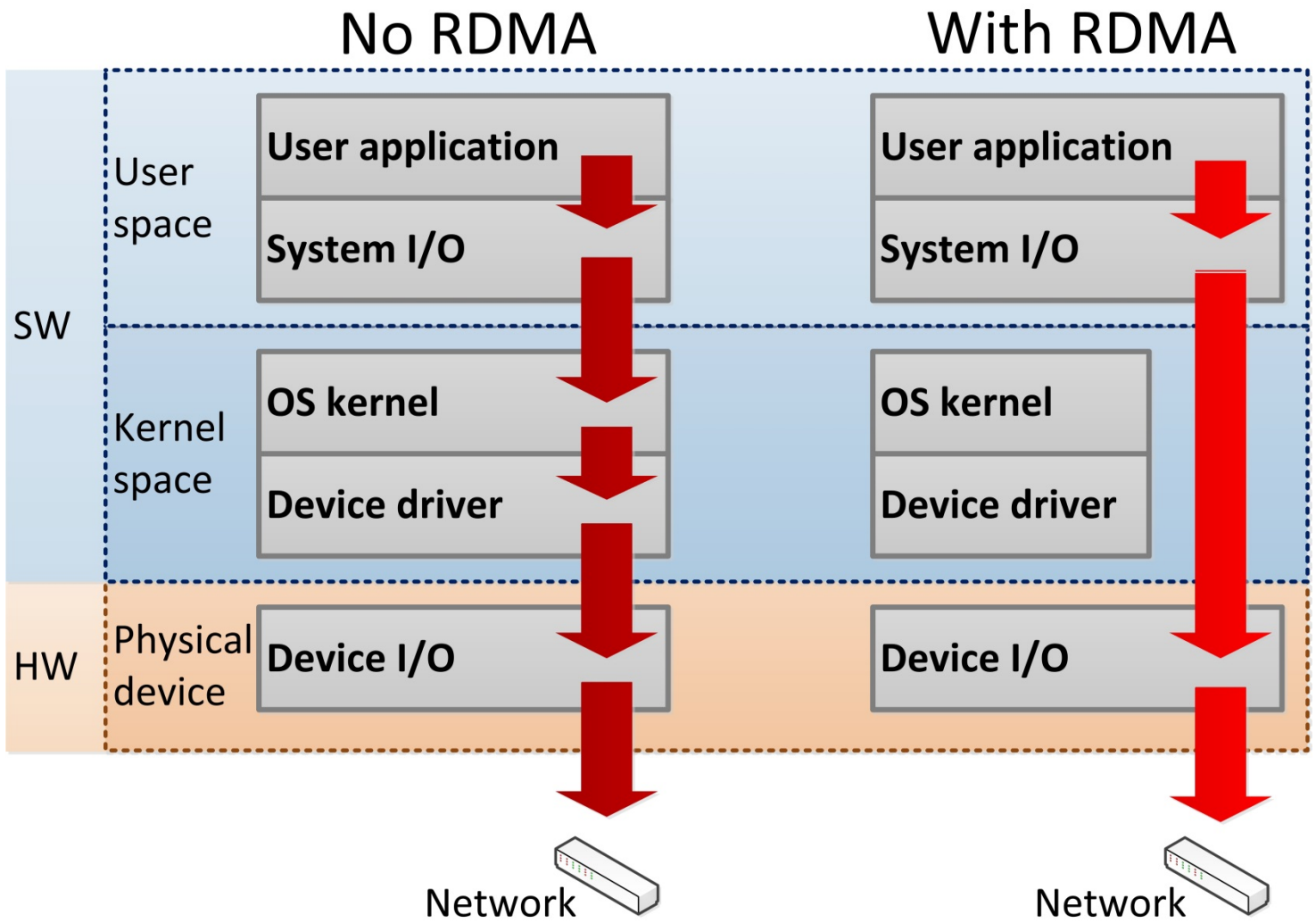
[1] ***StRoM**: Smart Remote Memory, EuroSys '20*
David Sidler, Zeke Wang, Monica Chiosa, Amit Kulkarni, Gustavo Alonso

ETH zürich Systems@ETH zürich DINFK

Farview

Why RDMA?

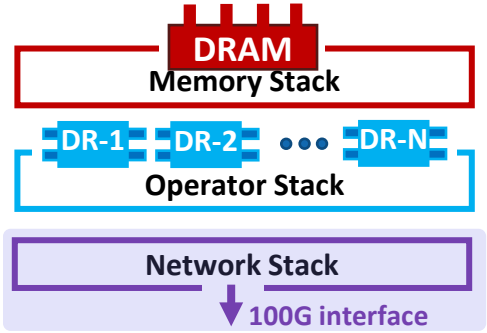
- Bypasses kernel space
- Zero-copy data movement
- Cheap pipelined processing (directly on the NIC)



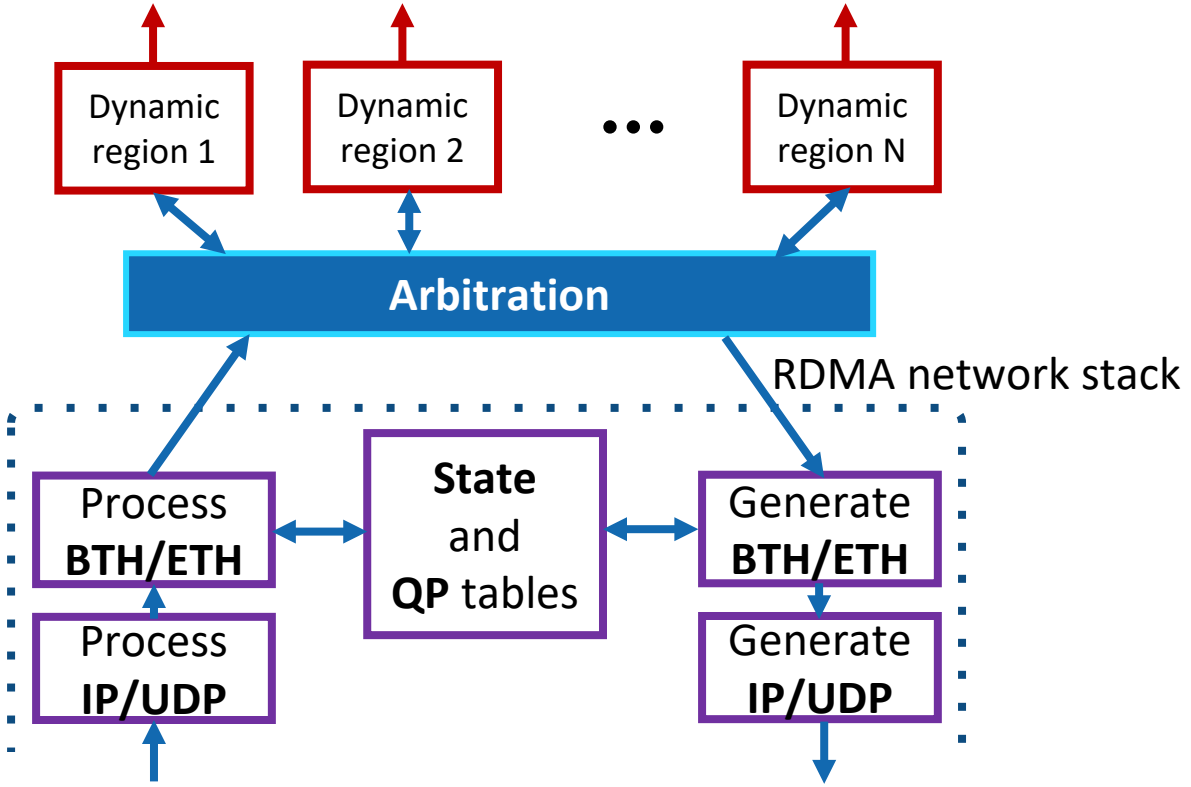
Farview

Network stack

- Manages all external connections and requests for all concurrent accesses
  - Supports ***RoCE v2*** at **100Gbps**
  - Open source network stack[1]
- Two-sided verbs used for invocation of queries by clients
- Comparable latencies to traditional one-sided RDMA verbs



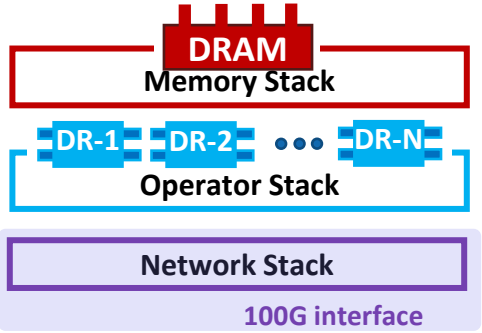
Network stack architecture:

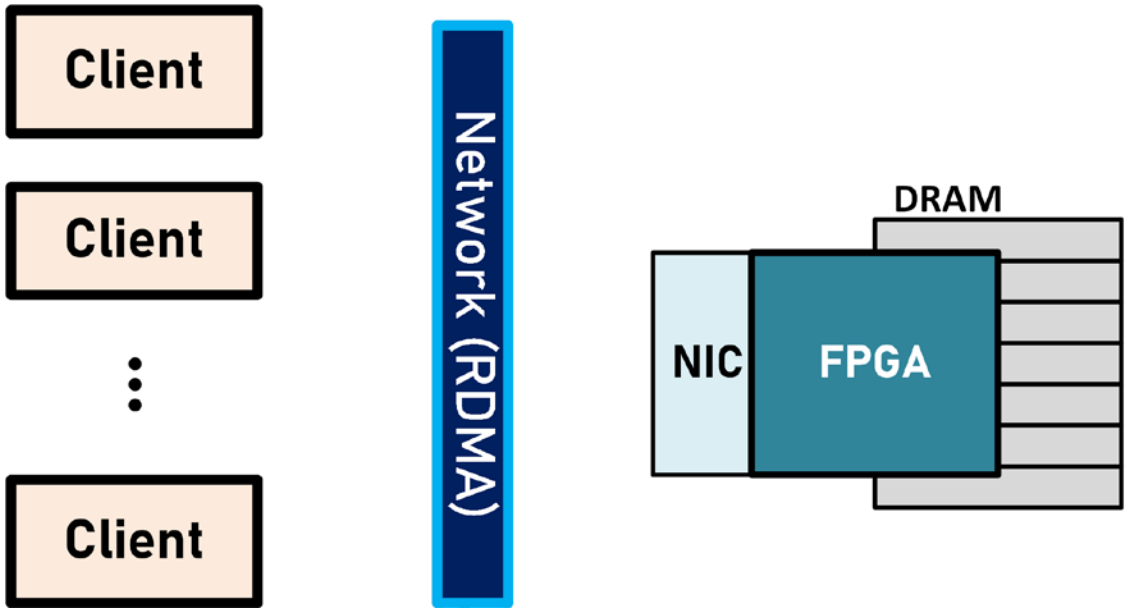


Farview

RDMA abstractions

- Remote memory abstraction (one sided verbs)
- Are current RDMA abstractions accurate for modern heterogeneous hardware?
- **Accelerator on the path, no longer just a memory access …**

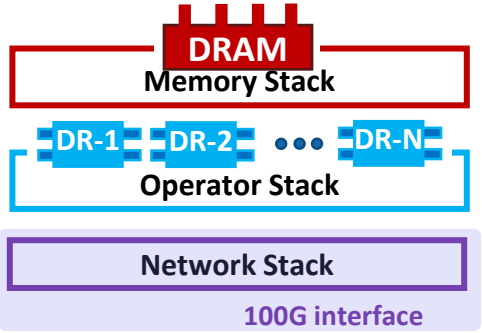


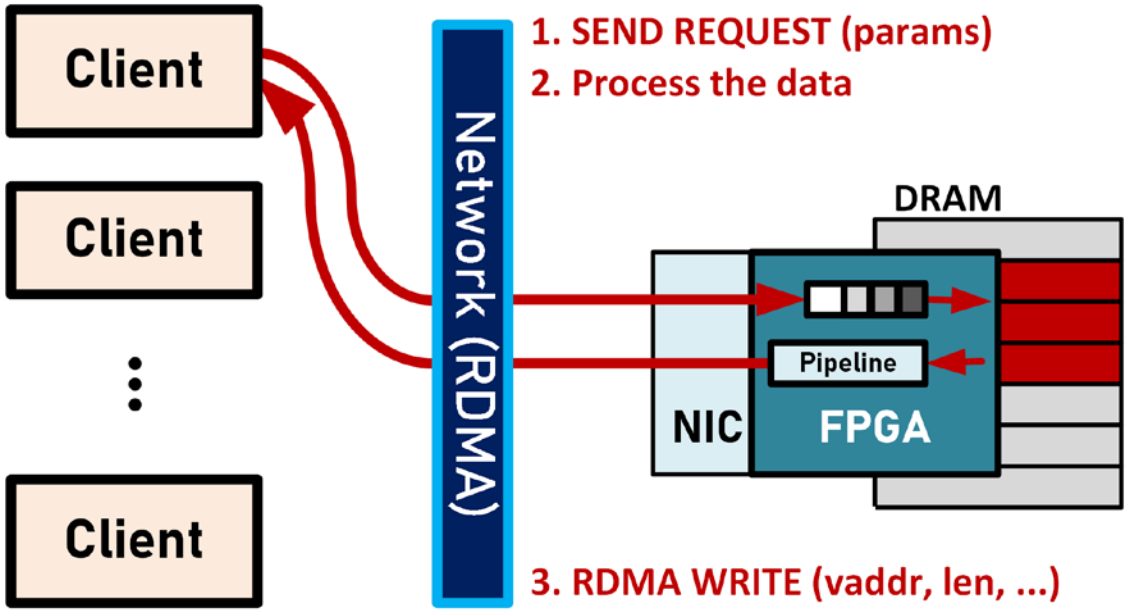


Farview

RDMA abstractions

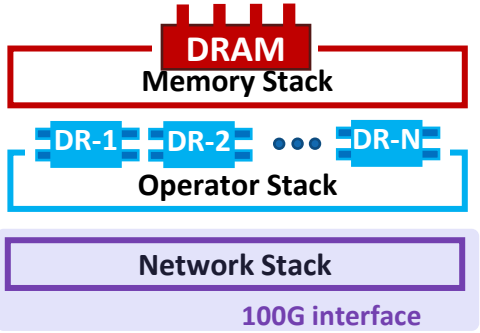
- Use existing IB verbs
- Performance of one-sided RDMA verbs (no overheads)
- Pass generic parameters (RPC)
- **Use SEND** (two-sided) **+ WRITE** (one-sided)





Farview

RDMA abstractions



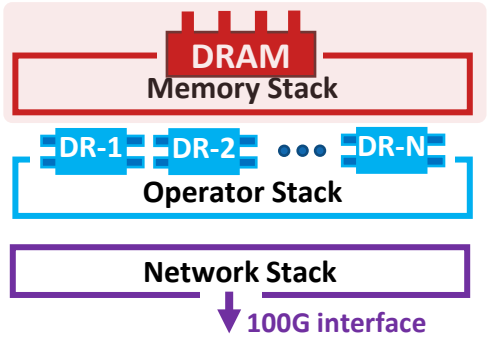
- **SEND** (two-sided) **+ WRITE** (one-sided)
- Pros:
  - Performance, latency comparable to traditional one-sided RDMA verbs
  - Generic enough (can pass as many parameters as needed)
- Cons:
  - Buffer information needs to be passed (could be evaded by extending the RDMA REQUEST verb instead)
  - Not a proper abstraction for what we are doing

- **A lightweight reliable layer on top of UDP?**

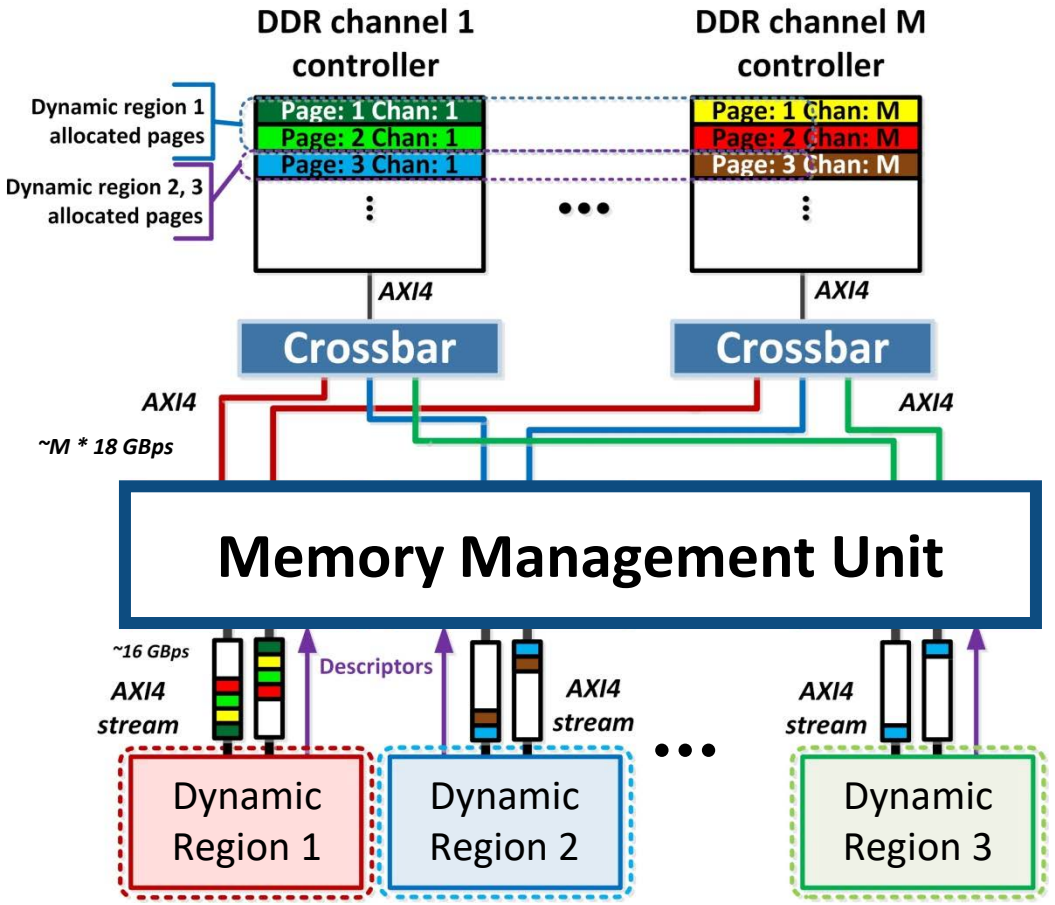
Farview

Memory stack

- Memory buffer pool
- Memory organized into multiple channels (interface same for DRAM and HBM)
- Striping abstraction to optimize the bandwidth
- Can process data at higher rates than the available network bandwidth
- **No host connections (PCIe …) overheads or bandwidth bottlenecks**

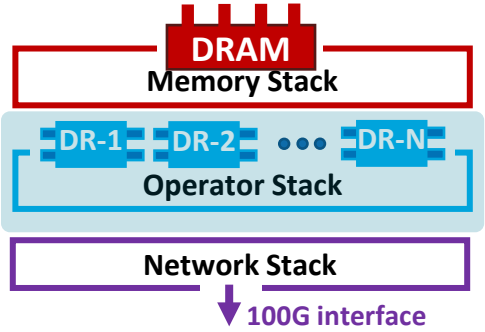


Memory stack architecture:



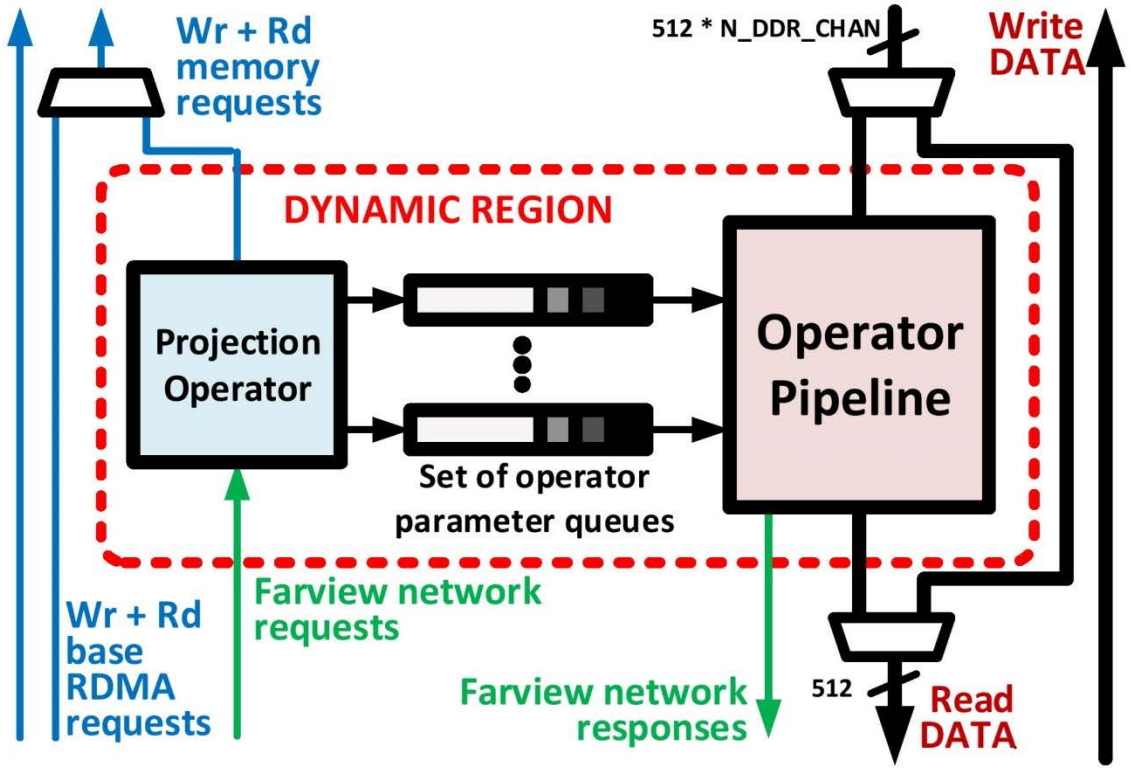
Farview

Operator stack



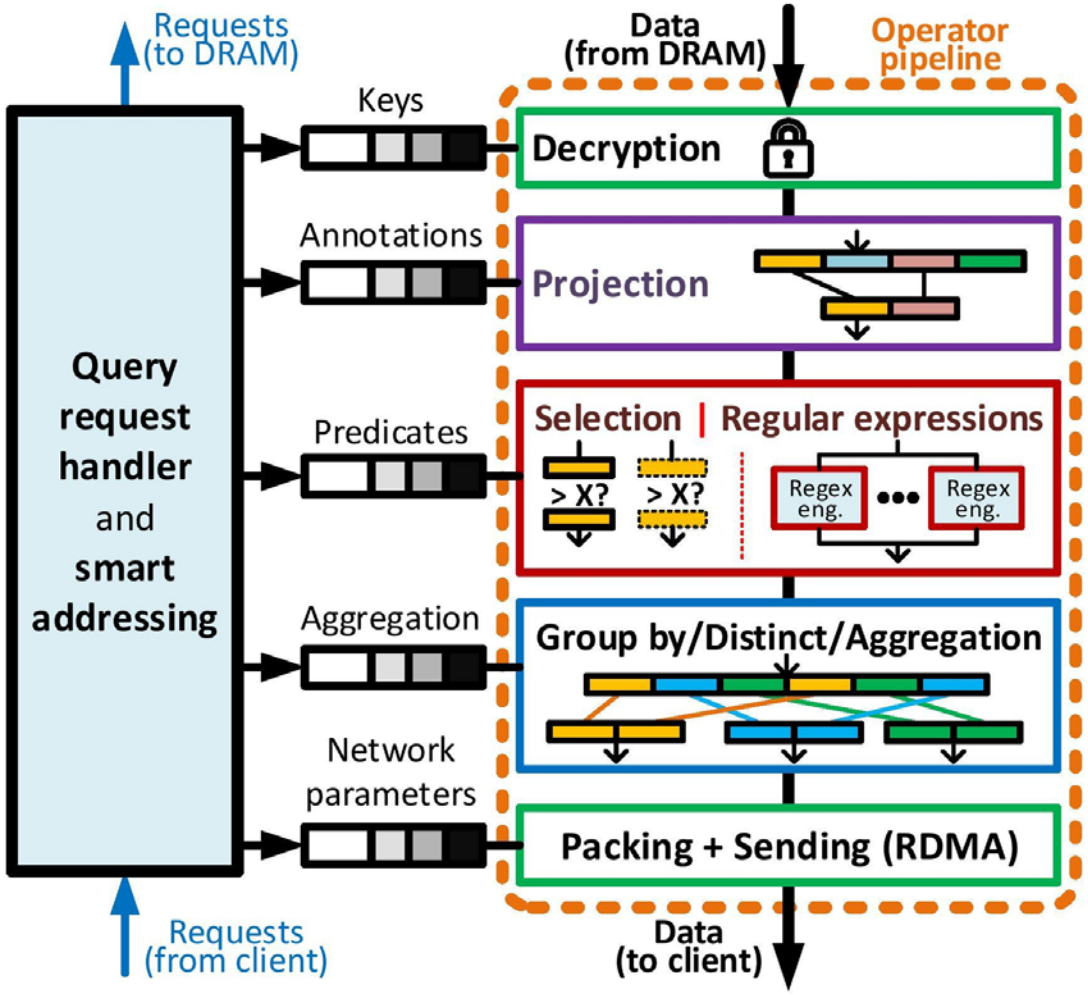
- **Operator stack** split into multiple isolated dynamic regions that operate concurrently (multiple clients)
- **Operator pipeline** can execute a range of queries
- **Operator pipelines** are swappable during runtime

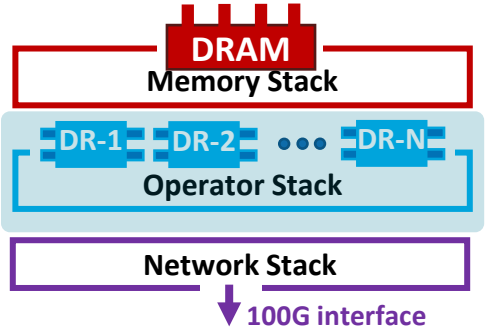
A single dynamic region and interfaces:



Farview

Operators

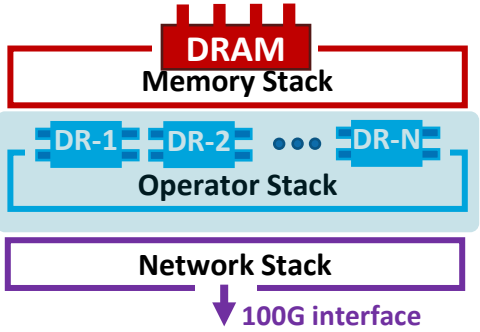




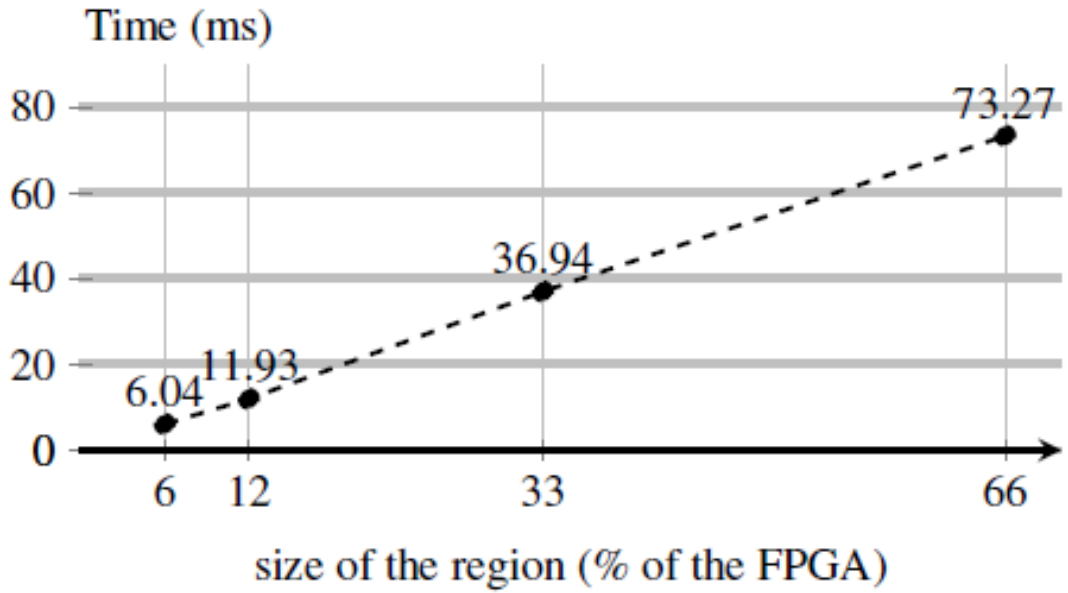
- An example of an *operator pipeline*
- Farview currently supports a variety of operators:
  - **Projection operators** (smart addressing, projection)
  - **Selection operators** (selection, regular expression matching, vectorized selection)
  - **Grouping operators** (distinct, group by and aggregation)
  - **System operators** (encryption/decryption, parsing, packing)
- Row store / column store
- Easily extendable

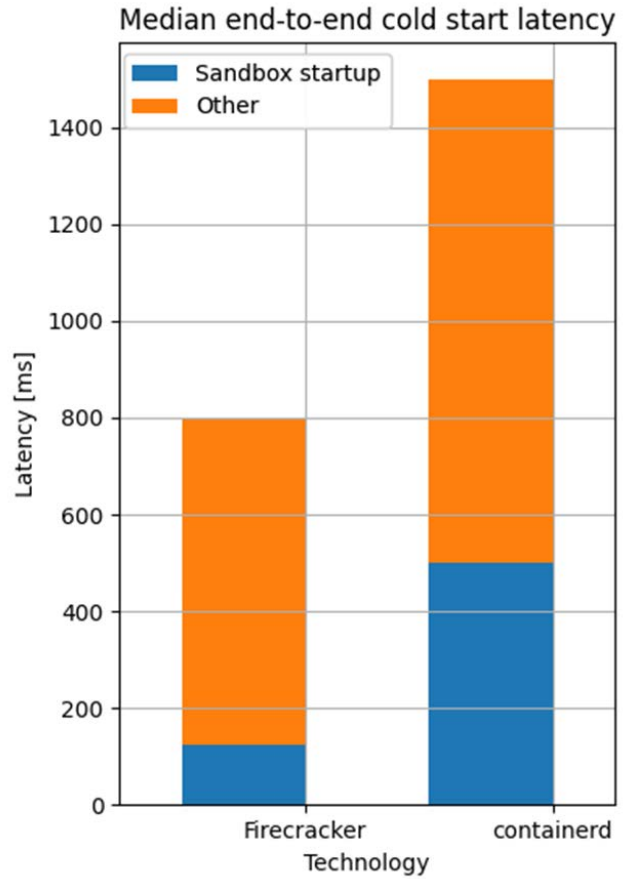
Farview

Operator Pipeline Swap



- Operator pipelines can be swapped during *runtime*
- Gives **Farview** a much needed flexibility in comparison to traditional accelerators
- Swap time in the order of milliseconds
- Could have applications in microservices domain





Farview

Programmatic Interface

- High level data API covering both the critical path operations and connection management operations
- API written in C++
- Intended to be used by Farview query compiler.

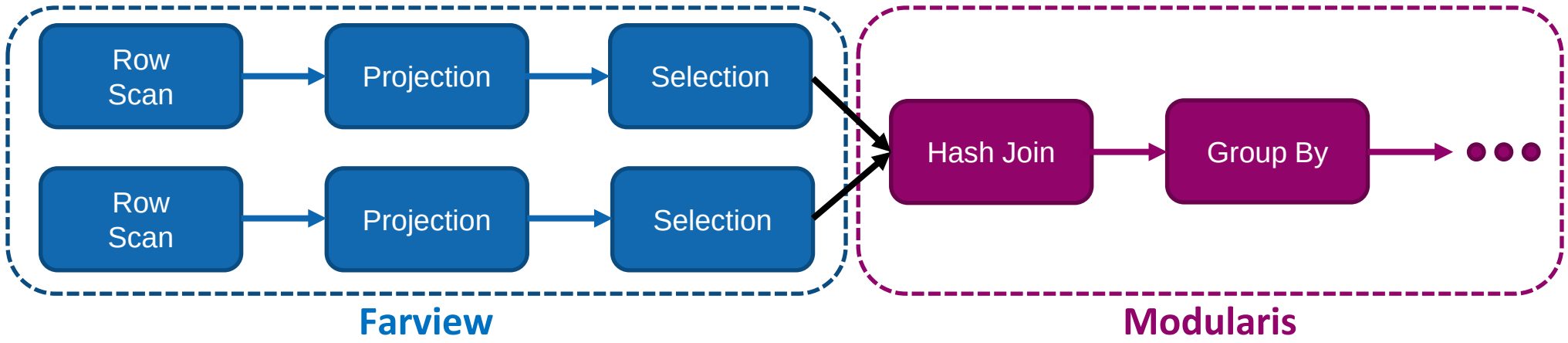
```
bool openConnection(Qpair *qp, Fview *node);
bool loadPipeline(Qpair *qp, int32_t opid);
void tableRead(Qpair *qp, Ftable *ft);
void tableWrite(Qpair *qp, Ftable *ft);
...
void farView(Qpair *qp, Ftable *ft, uint64_t *params);

void select(Qpair *qp, Ftable *ft, uint64_t *proj_flags,
uint64_t *sel_flags, float predicate) {
        ...
        farView(qp, ft, params);
}
```

Farview

Frontend - database engine

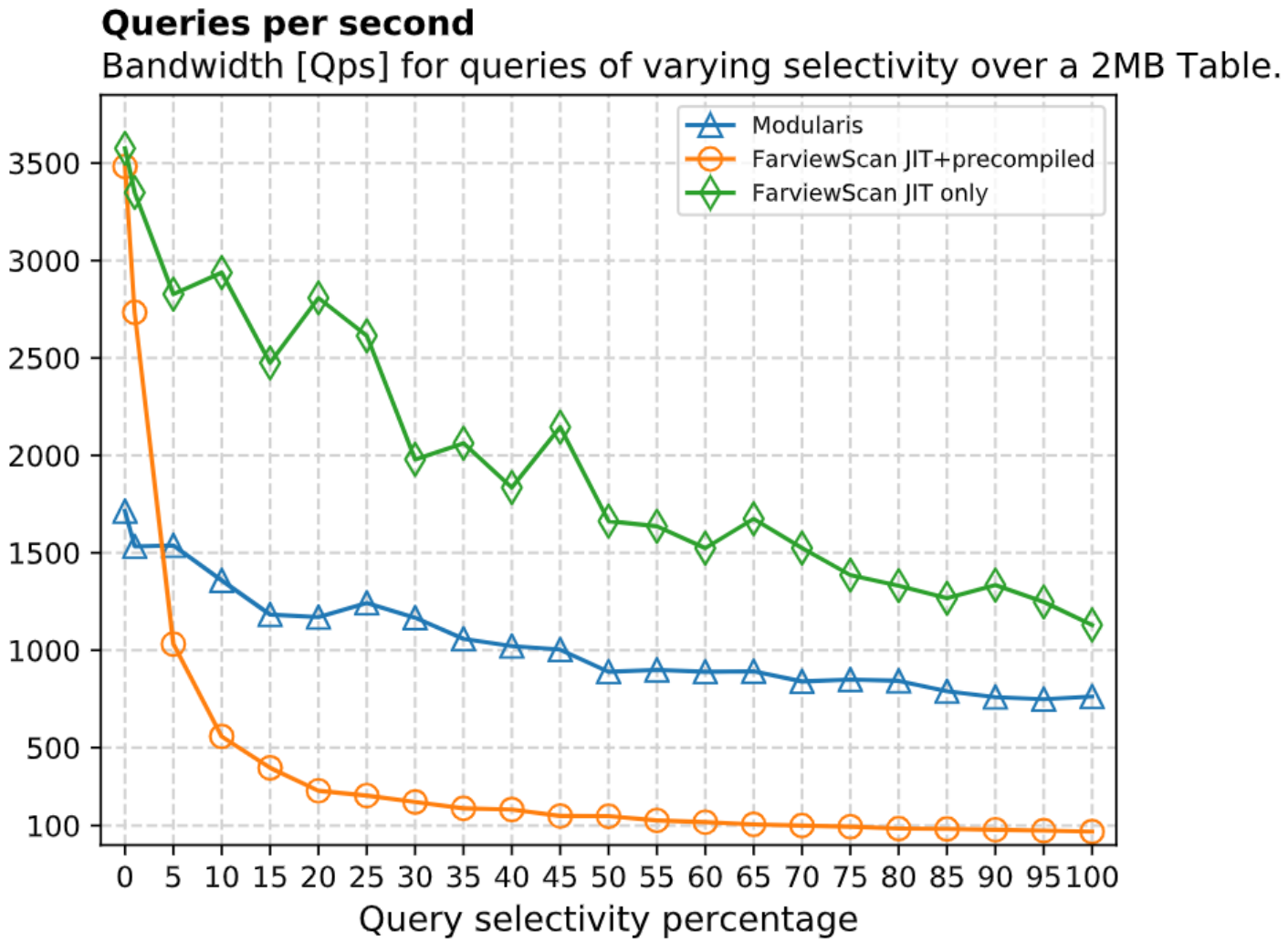
- How to interact with Farview from client on a higher level?
- **Modularis**[2] is a a distributed query processing system supporting relational queries with different backends (including RDMA)
- Offloading of certain operators within Modularis (Projections, Selections, Aggregations) to Farview



[2] ***Modularis**: Modular Relational Analytics over Heterogeneous Distributed Platforms, VLDB '20*
Dimitrios Koutsoukos, Ingo Müller, Renato Marroquín, Ana Klimovic, Gustavo Alonso

Farview

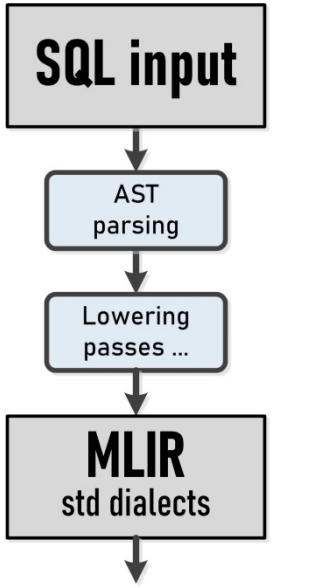
Frontend - database engine

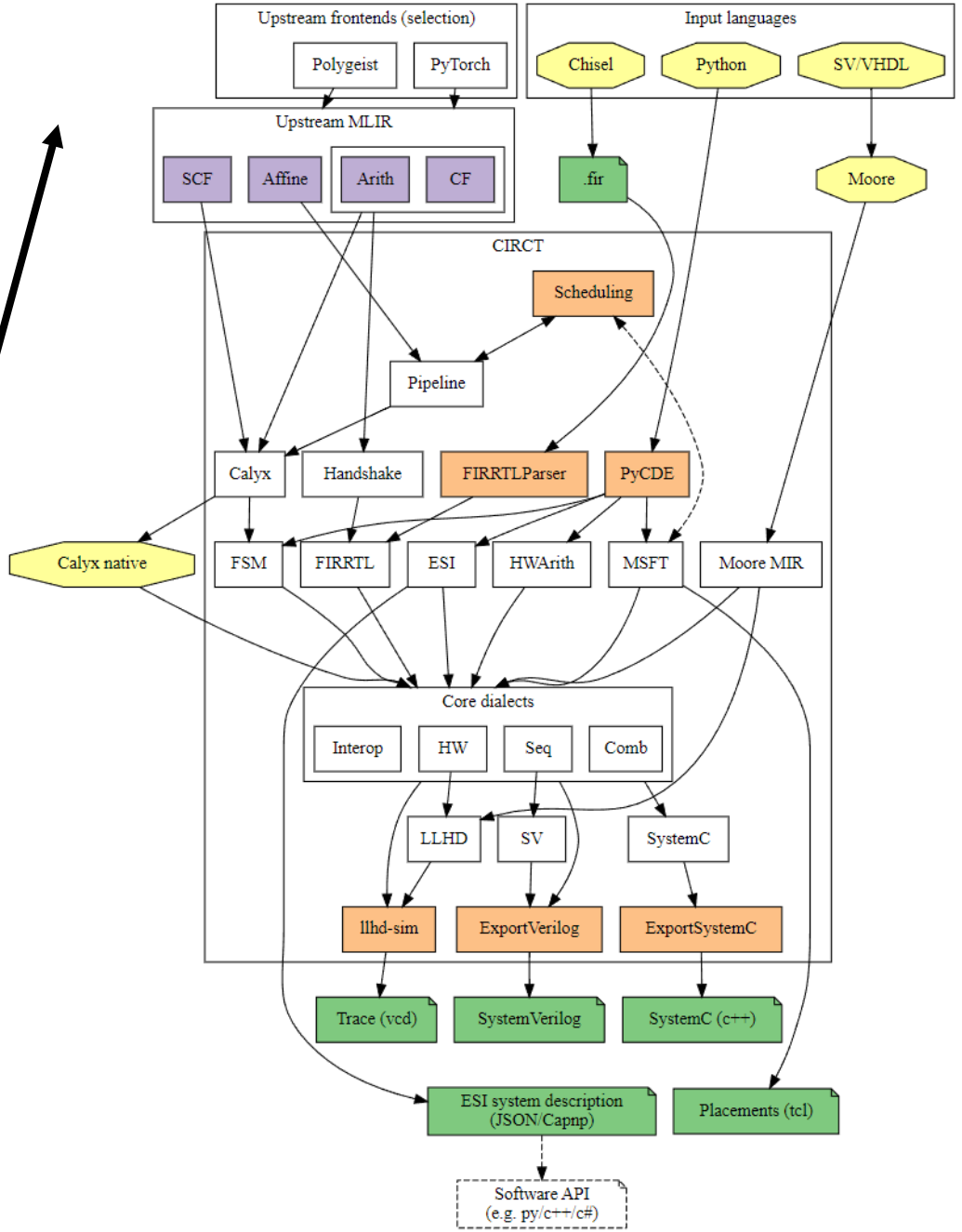


Farview

MLIR compiler

- How to create the operators?
- **MLIR** is a novel approach to building a compiler
- **CIRCT**[3] built within **MLIR**, produces **HDL**
- **Modularis** being ported to **MLIR**





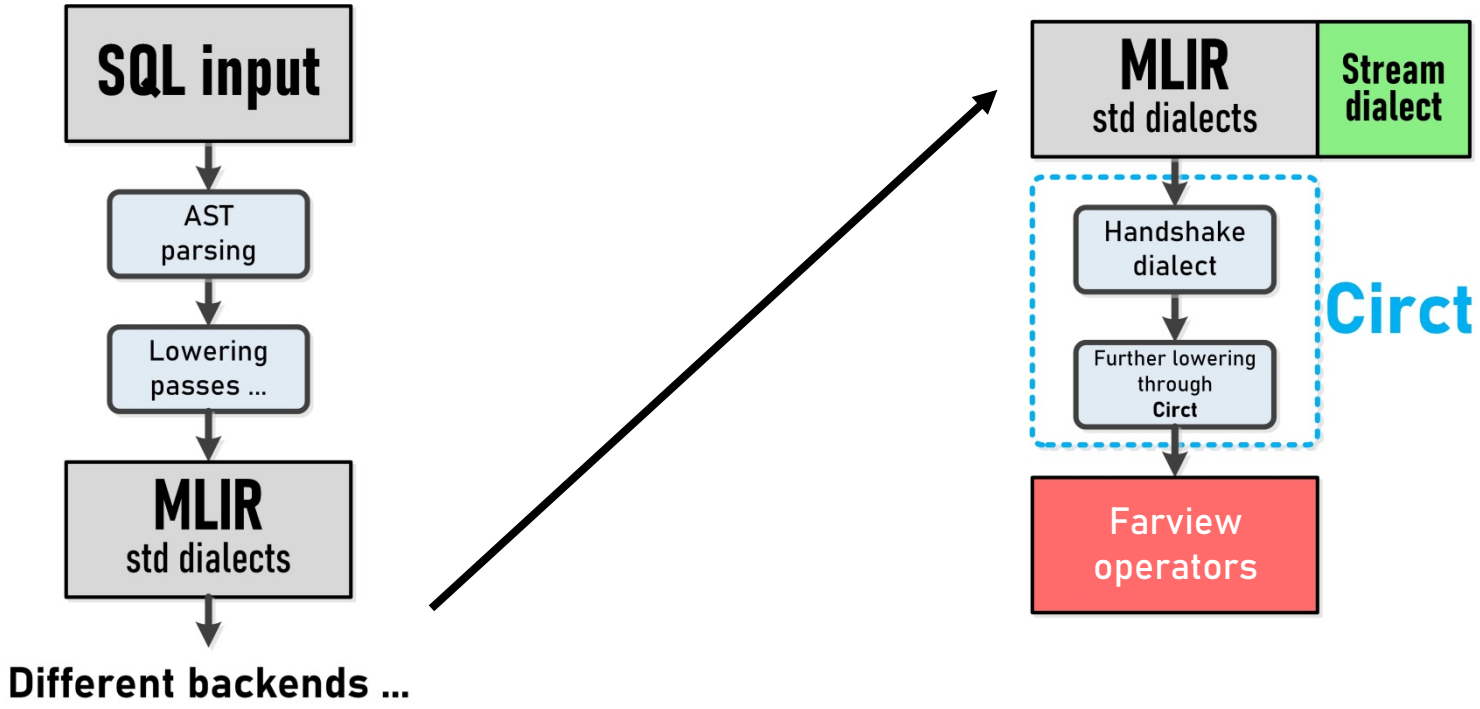
[3] *https://github.com/llvm/circt.git*

ETH zürich Systems@ETH zürich DINFK

Farview

Stream dialect

- Stream-dialect => Stream-CIRCT project



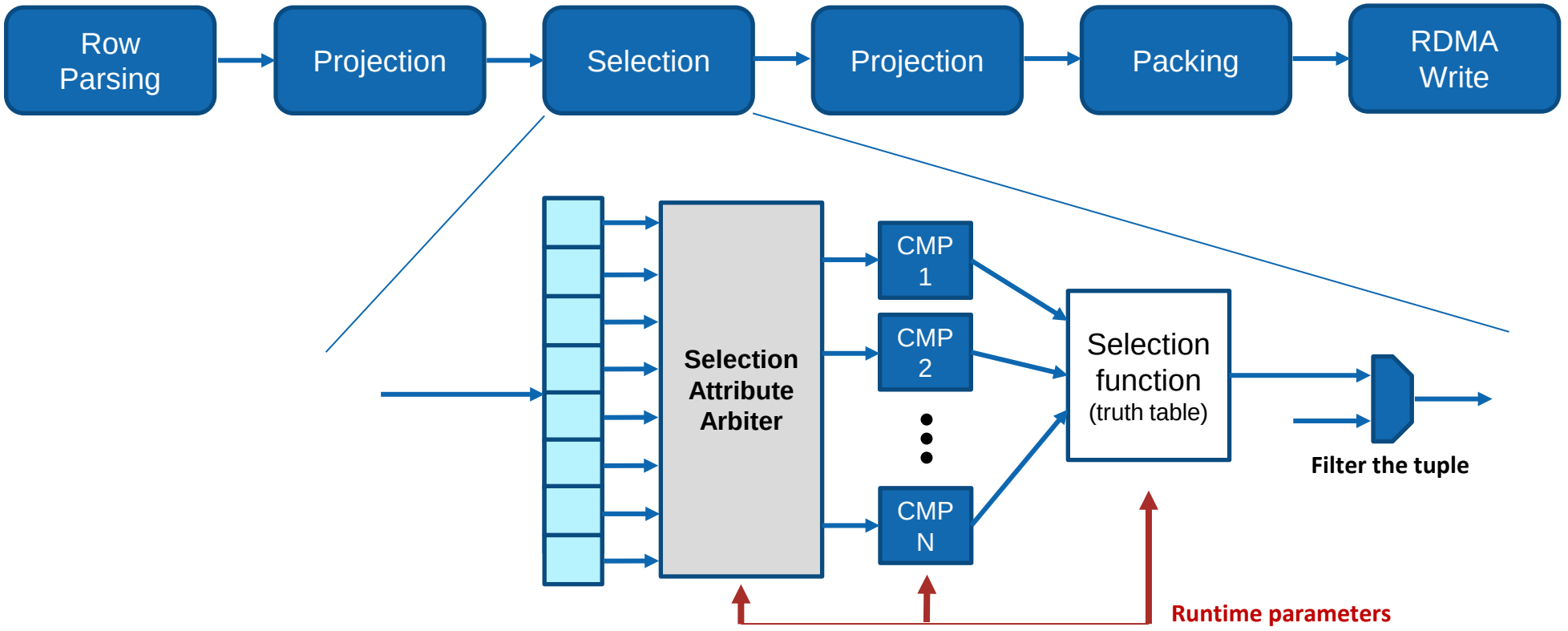
Evaluation

- **Farview** smart disaggregated buffer cache compared to two baselines:
  - (**LCPU**) Buffer cache implemented in local memory with processing on the local CPU
  - (**RCPU**) Remote buffer cache, no FPGA, implemented on a remote machine (Mellanox NIC)

- Benchmark comparisons to baselines performed for all implemented operators
  - RDMA throughput and response times microbenchmarks
  - Projection and smart addressing
  - Selection (100%, 50%, 25% selectivity)
  - Distinct queries
  - Group by queries
  - Regular expression matching
  - En/decryption
  - Multiple concurrent clients

Benchmarks (SELECT)

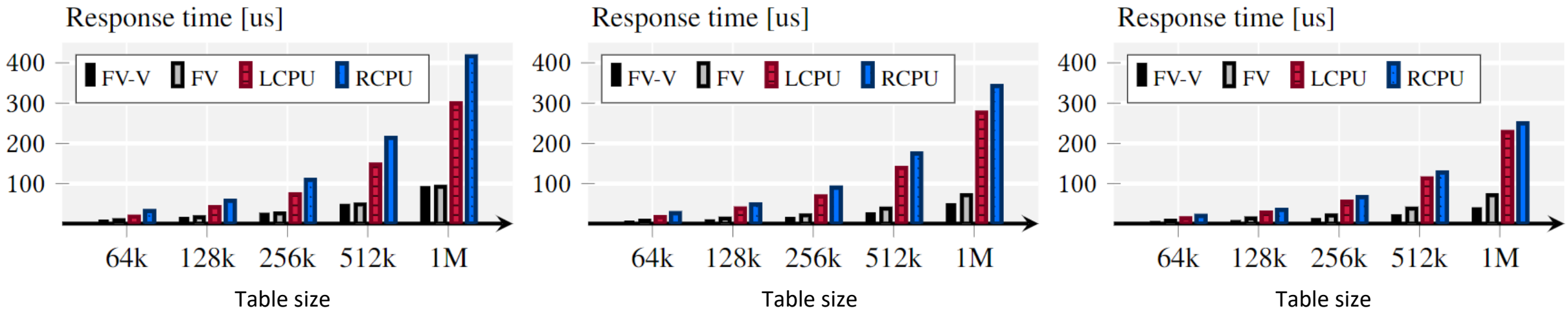
- Predicates provided at runtime
- Selection circuit generic enough for a wide range of selection queries



Benchmarks (SELECT)

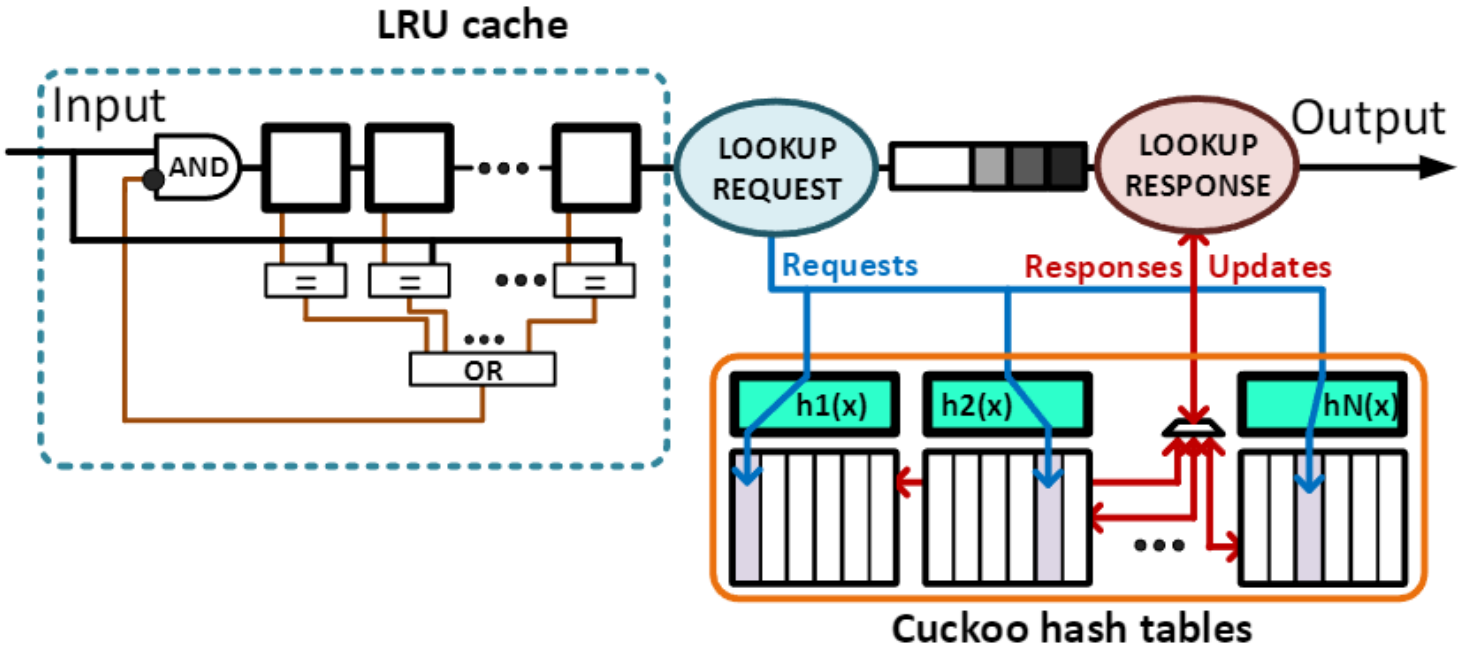
- **Farview** outperforms both baselines across different selectivity levels (due to the high filtering throughput of an FPGA with direct attached memory)
- Further increase in performance with parallelized pipelines (vectorization)

Response times for selection queries with 100%, 50% and 25 % selectivity, respectively:



Benchmarks (DISTINCT/GROUP BY)

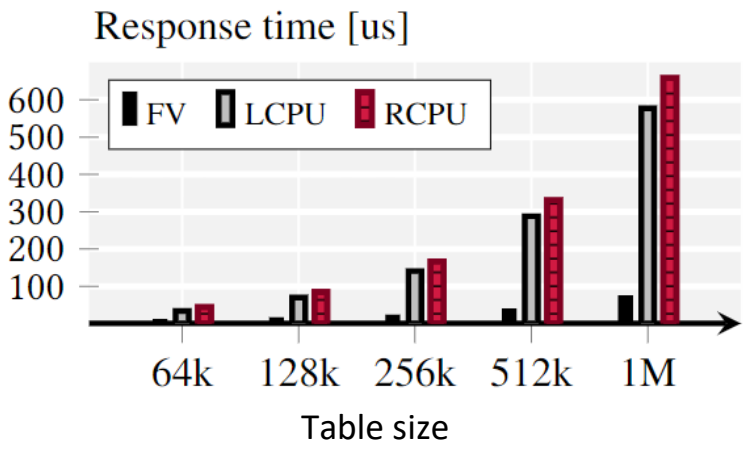
- Line-rate Distinct/Group by pipelines
- Cuckoo-hashing to reduce collisions
- No external bucket memory used (limited space in FPGA, might result in overflows)

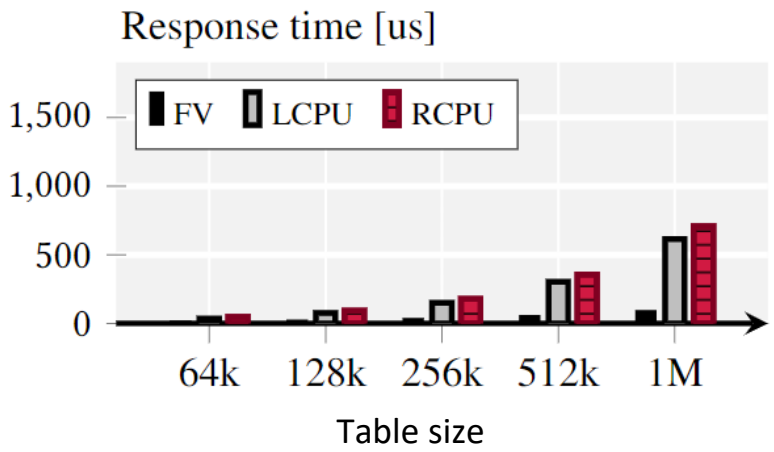


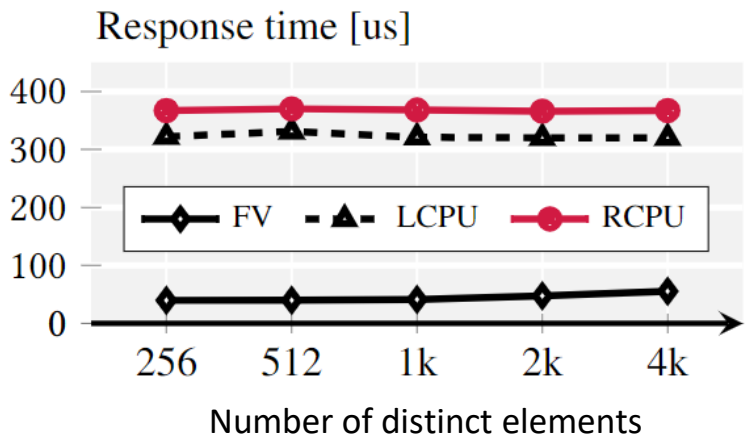
Benchmarks (DISTINCT/GROUP BY)

- **Farview** outperforms both baselines, advantage scales with the data size
- Tested scenarios where FPGA memory is sufficient
- In case of too many collisions, additional post processing needs to be done on the client side

Response time comparisons for a **distinct** query, a **group by** query, and a **group by** query by on a stable number of elements



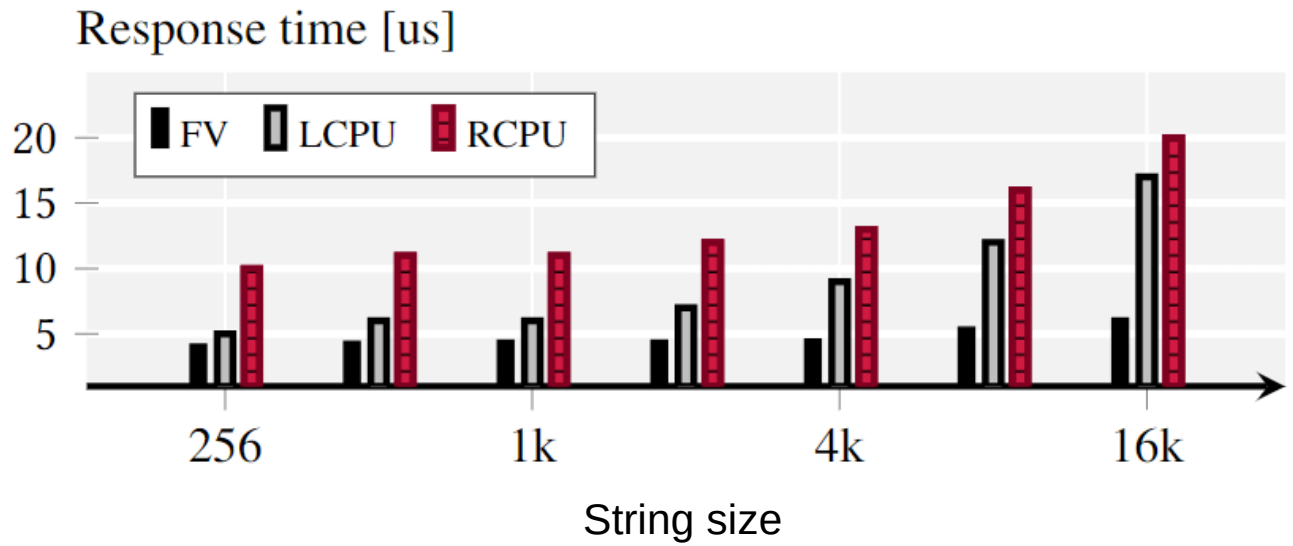




Benchmarks (REGEXP_LIKE)

- Regular expression matching is compute intensive in software
- Efficient compilation into NFAs on FPGAs[4]

Comparison of **REGEXP_LIKE** operator on different string lengths

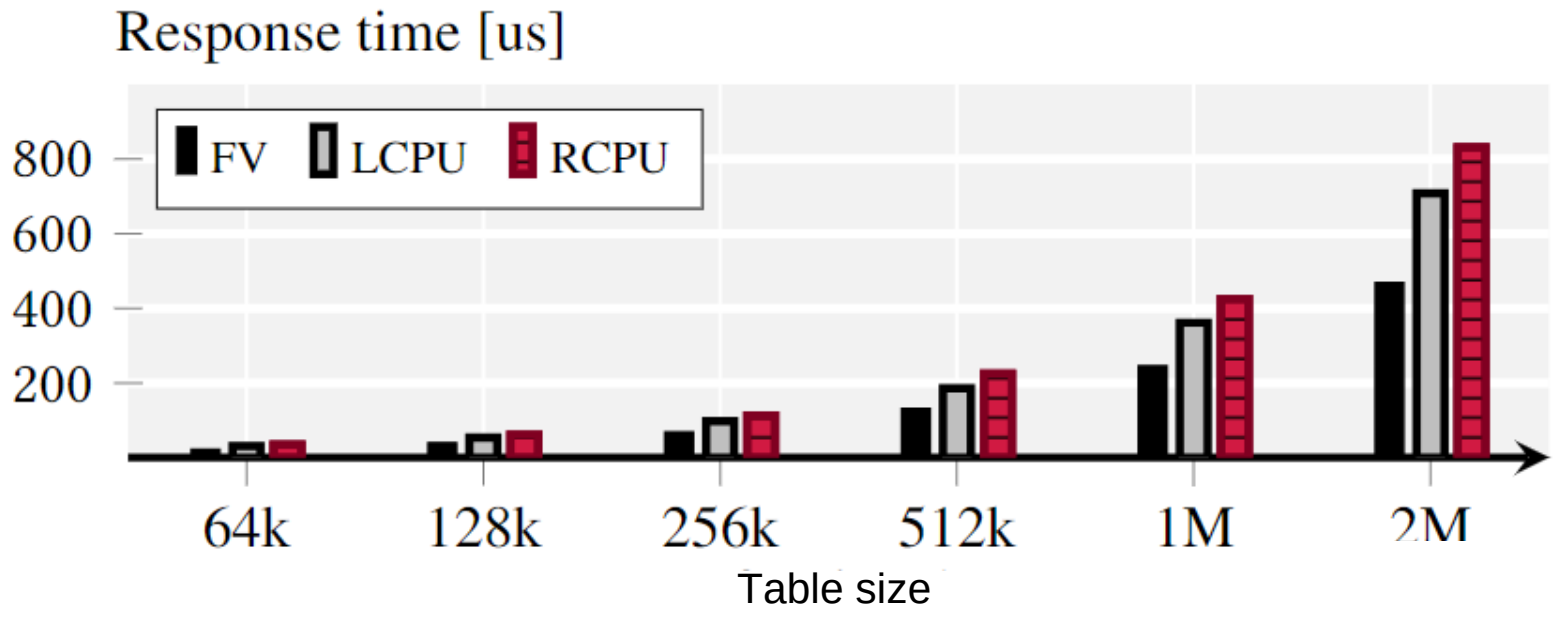


[4] *Runtime Parameterizable Regular Expression Operators for Databases, FCCM '16*
Zsolt István, David Sidler, Gustavo Alonso

Benchmarks (Concurrent Operators)

- Performance of concurrent operator pipelines (DISTINCT operation)
- Contention between multiple regions reduces the advantage of Farview

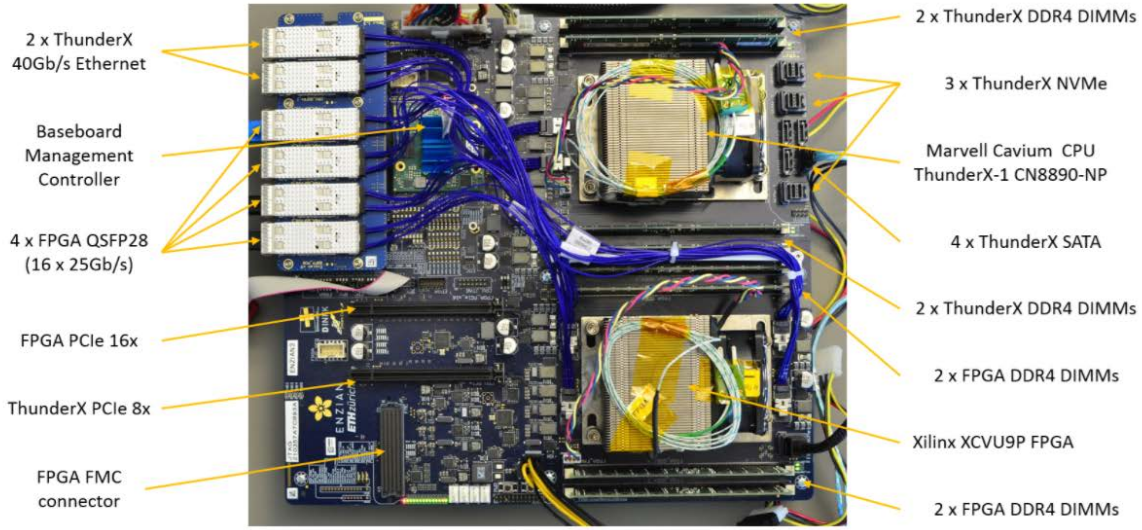
Performance comparison between 3 concurrent dynamic regions serving clients



Current and Future Work

- Implementation of the frontend for **Farview** *(Joining* **Farview** *and* **Modularis***)*
- **Farview** stream circuit compiler
- Implementation of a storage layer
- Extending the operator set (*joins …*)
- Distributed operation within the HACC cluster
- Larger scale deployments taking advantage of **Enzian** with 1 TB of DRAM per board:

ENZIAN



http://enzian.systems

Questions?