Accelerating Pattern Matching Queries in Hybrid CPU-FPGA Architectures

ETH *zürich*

David Sidler, Zsolt István, Muhsen Owaida, Gustavo Alonso
Systems Group, Department of Computer Science, ETH Zürich

Systems@ETH Zürich

New emerging hybrid architectures

Intel Xeon+FPGA (v1)
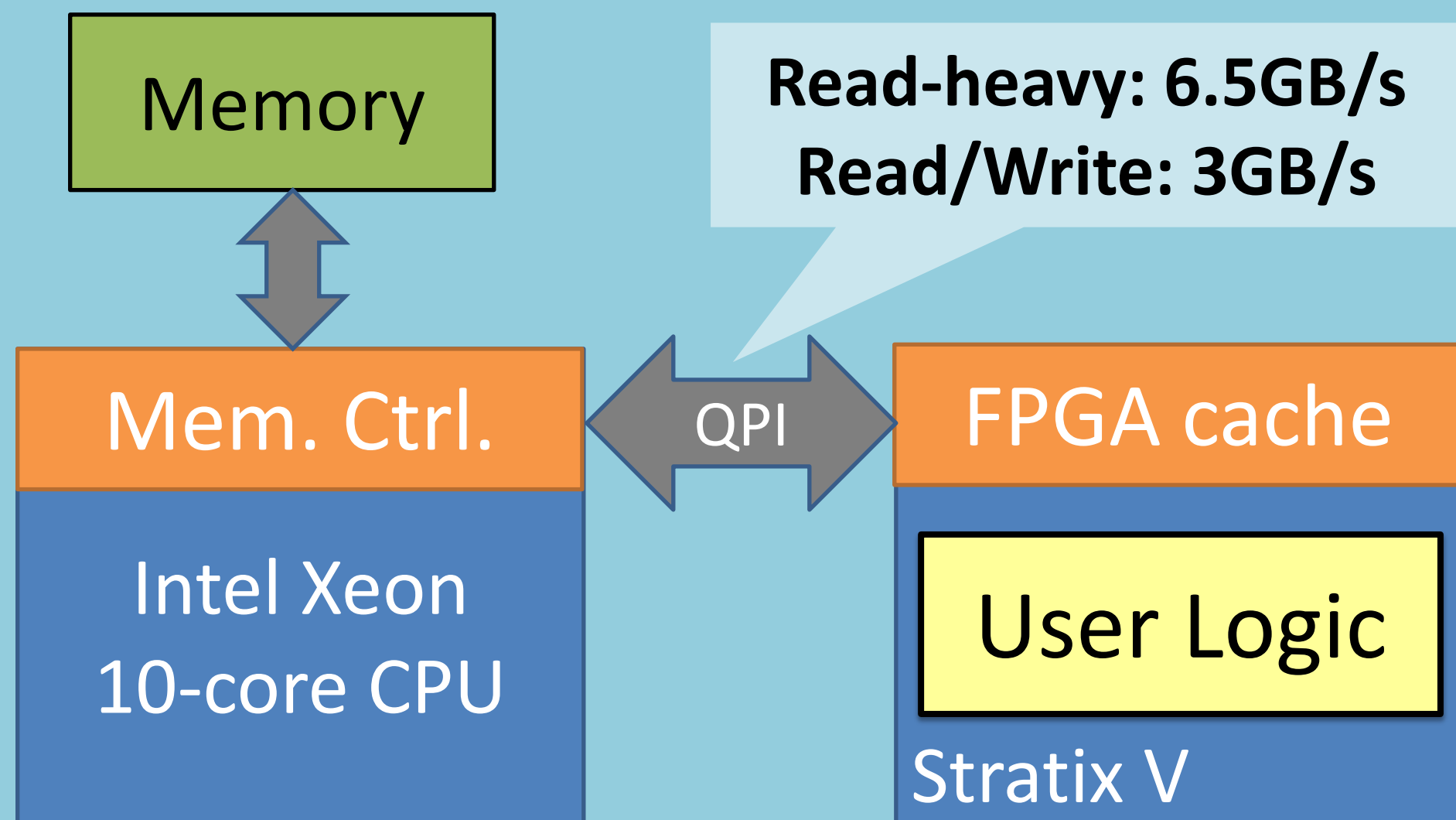
Memory

Read-heavy: 6.5GB/s
Read/Write: 3GB/s

Mem. Ctrl. ⟷ QPI ⟷ FPGA cache

Intel Xeon 10-core CPU

User Logic

Stratix V

Hybrid CPU-FPGA Architectures:
• FPGA has cache-coherent access to main memory
• Eliminate the issue of data movement/partitioning
• FPGA is a **specialized** core

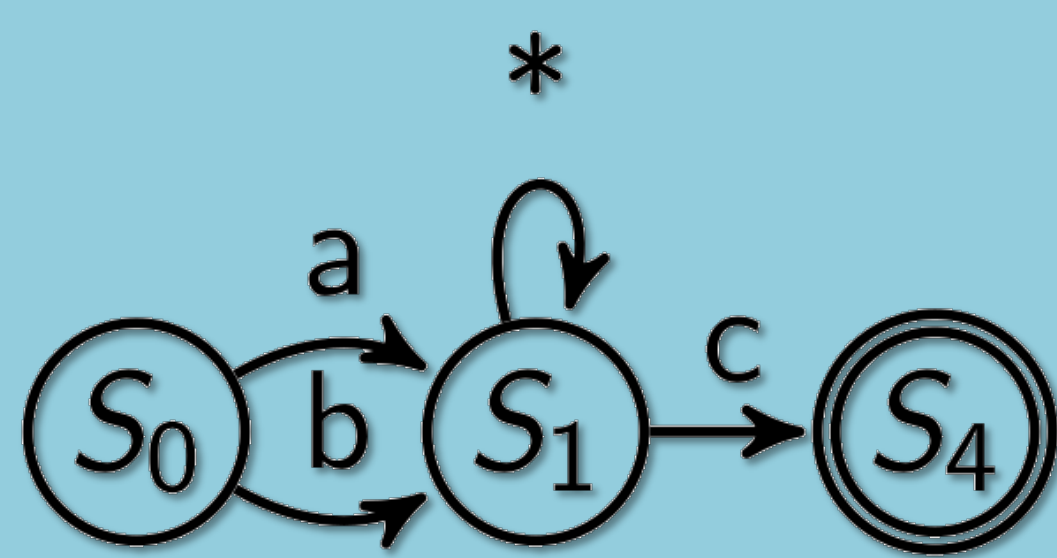*Your next CPU might come with an FPGA!*

Intel Xeon+FPGA (v2)

intel XEON inside™ ➕ intel ARRIA inside™

CPU and FPGA in same package

How to use your future FPGA for pattern matching

Regular Expression in Hardware:
• Regular expressions can be mapped to NFAs
• NFAs can be efficiently executed on FPGAs[1,2]

Regular Expression: (a|b).*c

S_0 —a,b→ S_1 (*) —c→ S_4

[1] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs,* FCCM'01
[2] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

Character Encoder supports:
• Sequences • Case-insensitivity
• Ranges • Collations

Transitions parameterized by State Transitions matrix

Characters configuring Character Encoder

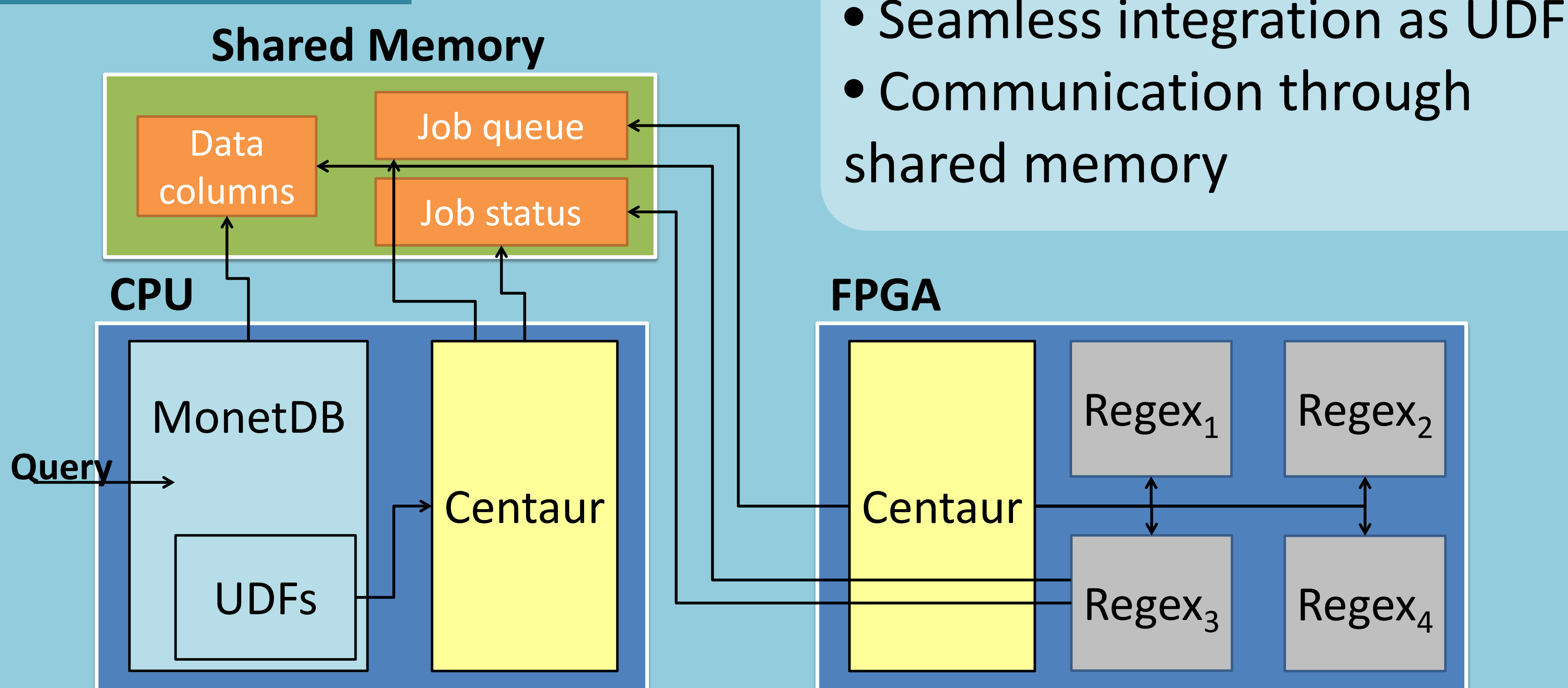**Characters triggering states**

Input character → Character Encoder

Characters			
C1	C2	C3	C4
'a'	'b'	'c'	

Triggers	C1	C2	C3	C4
S1	1	1	0	0
S2	0	0	0	0
S3	0	0	1	0
S4	0	0	1	0

C1 C2 C3 C4 → S1 S2 S3 S4

State Transitions	S1	S2	S3	S4
S1	1	0	0	0
S2	0	0	0	0
S3	0	0	0	0
S4	1	0	0	0

State Graph (fully connected)

Integration

Shared Memory

Data columns
Job queue
Job status

CPU
MonetDB
UDFs
Centaur

Query →

FPGA
Centaur
Regex$_1$ Regex$_2$
Regex$_3$ Regex$_4$
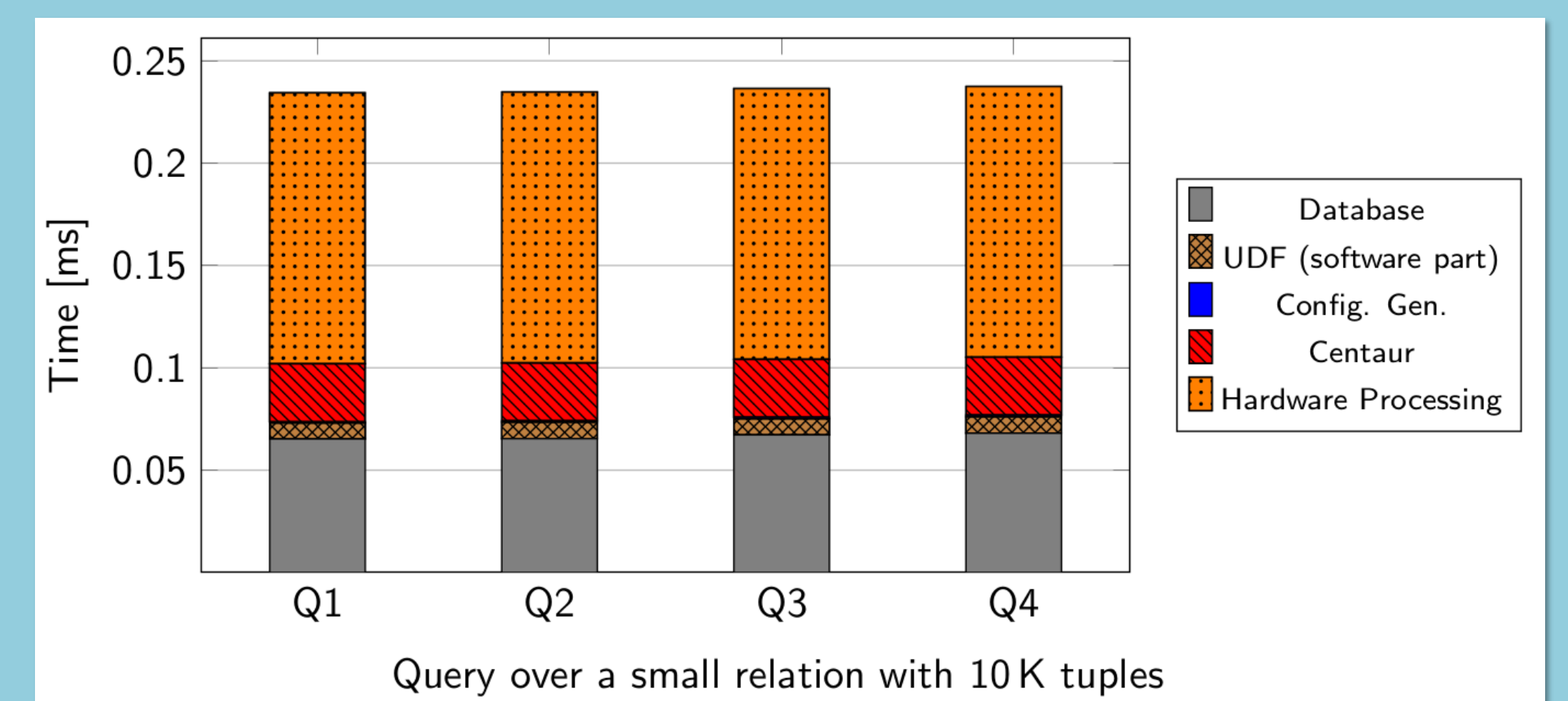
• Seamless integration as UDF
• Communication through shared memory

[3] M. Owaida, D. Sidler, et al., *Centaur: A Framework for Hybrid CPU-FPGA Databases*, FCCM'17

Overhead

Query over a small relation with 10 K tuples

Legend: Database, UDF (software part), Config. Gen., Centaur, Hardware Processing

Minimal overhead which decreases proportionally for larger data sets

Throughput

MonetDB — FPGA
Number of Clients ; Queries/s ; Q_1 ; $Q_{1,2,3,4}$; Q_4 ; Q_2 ; Q_3

DBx — FPGA
Number of Clients ; Queries/s ; $Q_{1,2,3,4}$; Q_1 ; Q_3 ; Q_2 ; Q_4
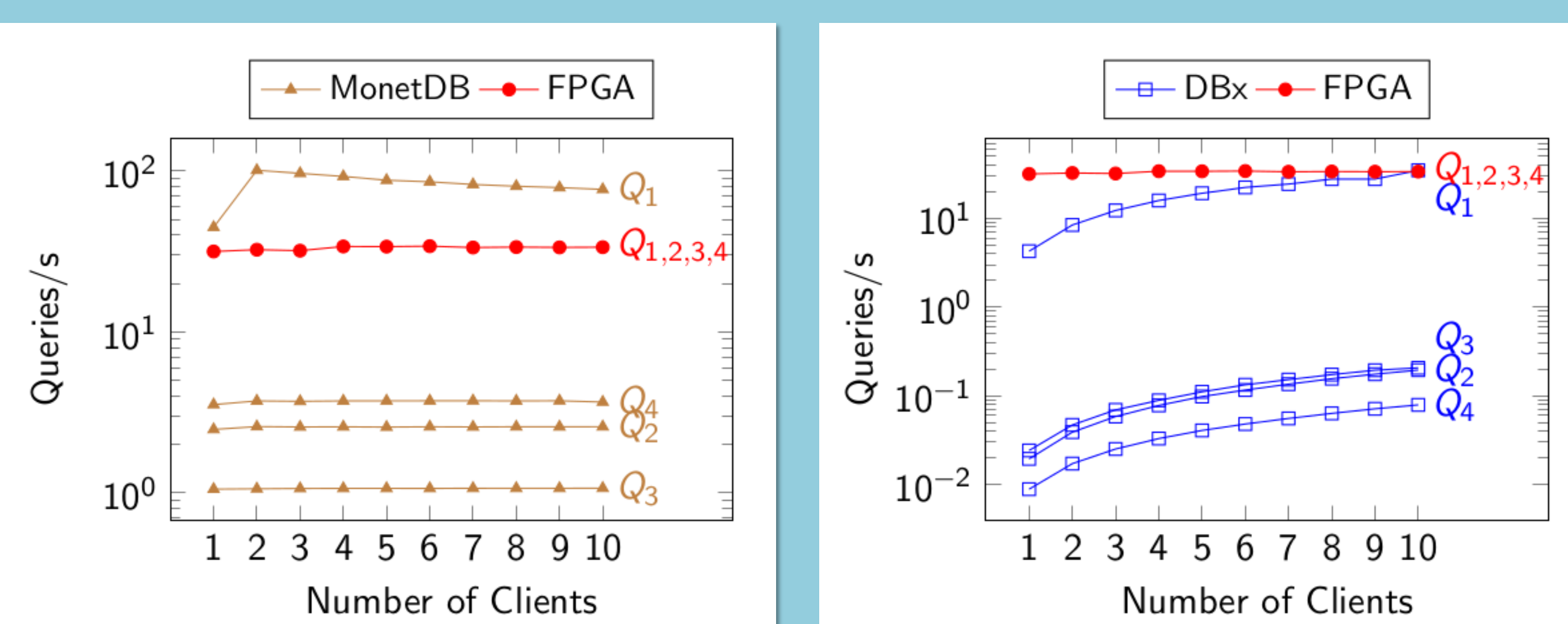
On a par with a 10-core CPU!

Hybrid Execution

• Complex expressions might not fit into deployed NFA
• Divide regex evaluation between CPU and FPGA

MonetDB — Hybrid UDF
Selectivity (also, fraction post-processed on CPU) ; Queries/s

systems.ethz.ch/fpga
github.com/fpgasystems