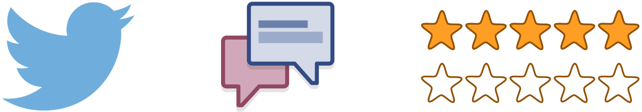
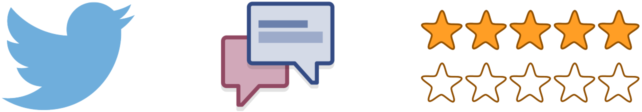
Accelerating Pattern Matching Queries in Hybrid CPU-FPGA Architectures

David Sidler, Zsolt István, Muhsen Owaida, Gustavo Alonso
Dept. of Computer Science, ETH Zürich

Increasing amount of user generated data

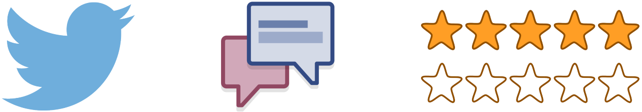


Increasing amount of user generated data



Query (WHERE clause)	Response time (s)	
Database	MonetDB	DBx
`LIKE '%Alan%Turing%Cheshire%'`	0.02	0.43
`REGEXP_LIKE('Alan.*Turing.*Cheshire')`	0.36	8.86

Increasing amount of user generated data

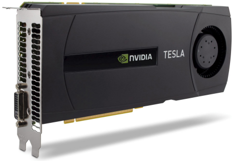


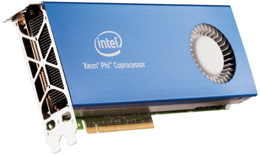
Query (WHERE clause)	Response time (s)	
Database	MonetDB	DBx
`LIKE '%Alan%Turing%Cheshire%'`	0.02	0.43
`REGEXP_LIKE('Alan.*Turing.*Cheshire')`	0.36	8.86

Databases are not suitable for complex text queries!

Accelerators to the rescue

- Using GPUs [1,2] or Xeon Phi [3] to accelerate string matching:
  - High speed-up
  - Data already on accelerator or data movement reduces acceleration benefit
  - Change of data layout
  - Performance depends on pattern complexity

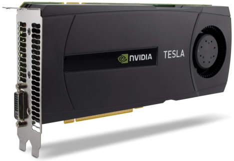


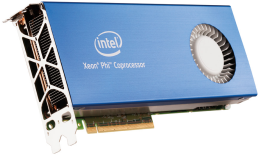


[1] E. Sitaridi, K. Ross, *GPU-Accelerated string matching for database applications*, VLDB Journal, Oct. 2016
[2] C.-H. Lin, et al., *Accelerating regular expression matching using hierarchical parallel machines on GPU*, GLOBECOM'11
[3] E. Sitaridi, O. Polychroniou, K. Ross, *SIMD-Accelerated regular expression matching*, DAMON'16

Accelerators to the rescue

- Using GPUs [1,2] or Xeon Phi [3] to accelerate string matching:
  - High speed-up
  - Data already on accelerator or data movement reduces acceleration benefit
  - Change of data layout
  - Performance depends on pattern complexity
- Integration into database engine often unclear

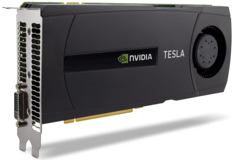


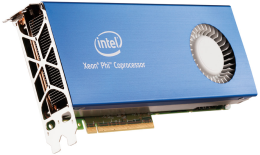


[1] E. Sitaridi, K. Ross, *GPU-Accelerated string matching for database applications*, VLDB Journal, Oct. 2016
[2] C.-H. Lin, et al., *Accelerating regular expression matching using hierarchical parallel machines on GPU*, GLOBECOM'11
[3] E. Sitaridi, O. Polychroniou, K. Ross, *SIMD-Accelerated regular expression matching*, DAMON'16

Accelerators to the rescue

- Using GPUs [1,2] or Xeon Phi [3] to accelerate string matching:
  - High speed-up
  - Data already on accelerator or data movement reduces acceleration benefit
  - Change of data layout
  - Performance depends on pattern complexity
- Integration into database engine often unclear



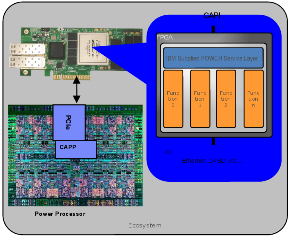


Data partitioning/movement hinders wide-spread adoption of database accelerators!

[3] E. Sitaridi, O. Polychroniou, K. Ross, *SIMD-Accelerated regular expression matching*, DAMON'16

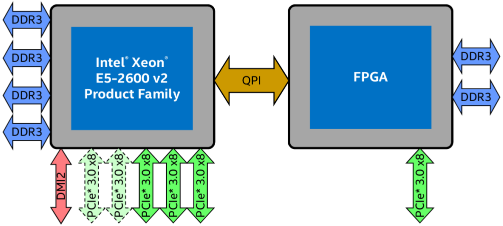
New hybrid architectures are emerging

IBM Power8 + CAPI



Source: Heterogeneous computing on POWER, Cesar Diniz Maciel, IBM

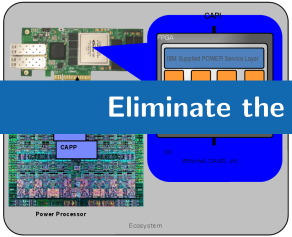
Intel Xeon+FPGA



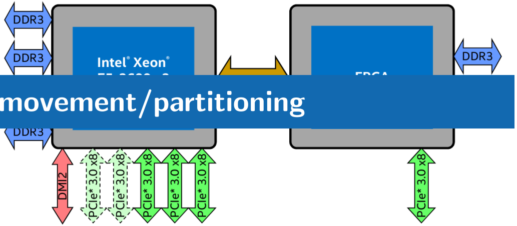
Source: Intel Xeon+FPGA Platform for the Data Center, PK Gupta, Intel

New hybrid architectures are emerging

IBM Power8 + CAPI



Intel Xeon+FPGA



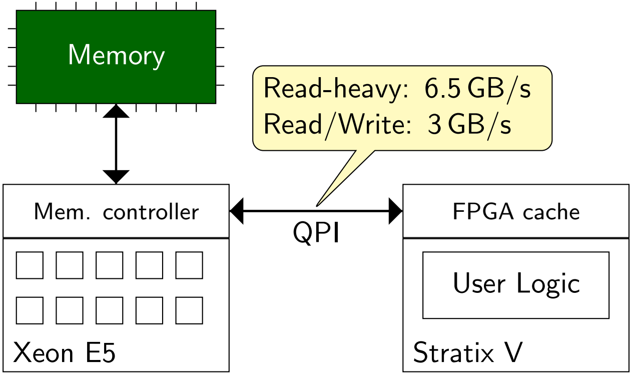
Eliminate the issue of data movement/partitioning

Source: Heterogeneous computing on POWER, Cesar Diniz Maciel, IBM

Source: Intel Xeon+FPGA Platform for the Data Center, PK Gupta, Intel

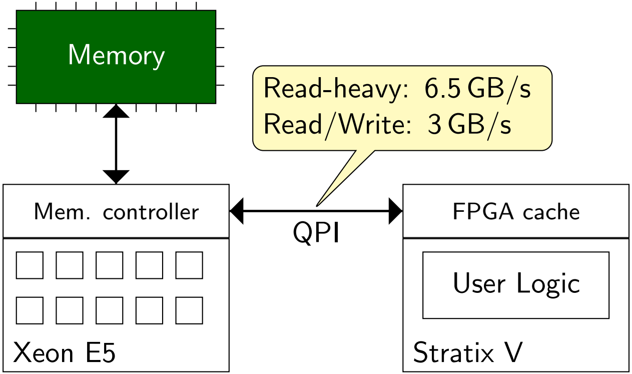
Intel Xeon+FPGA prototype platform

Version 1 (used in this work)

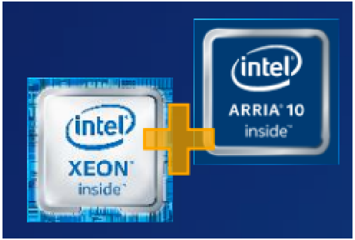


Intel Xeon+FPGA prototype platform

Version 1 (used in this work)



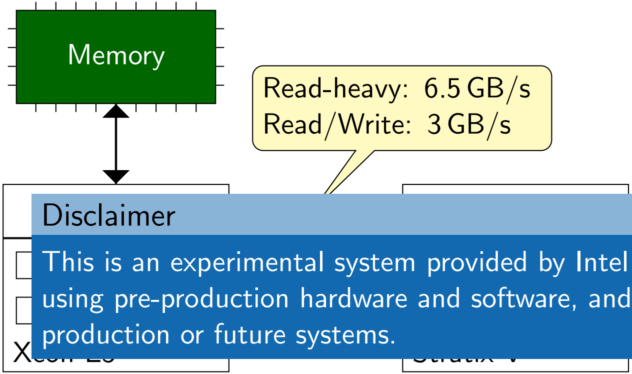
Version 2



- Larger bandwidth (1xQPI, 2xPCI)
- Larger FPGA
- FPGA in same package (single socket)

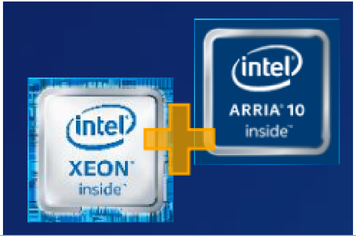
Intel Xeon+FPGA prototype platform

Version 1 (used in this work)



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

Version 2

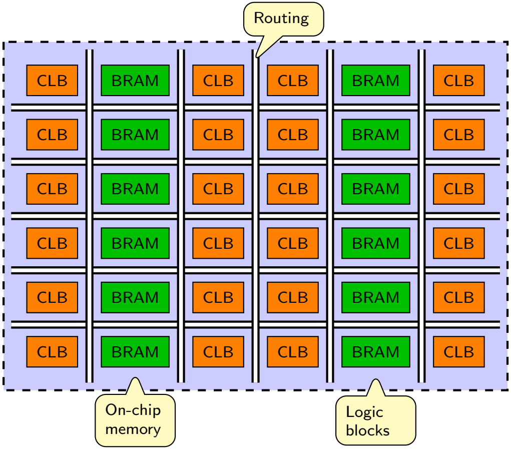


- FPGA in same package (single socket)

Disclaimer

This is an experimental system provided by Intel any results presented are generated using pre-production hardware and software, and may not reflect the performance of production or future systems.

FPGA (Field Programmable Gate Array)


CLB BRAM CLB CLB BRAM CLB
CLB BRAM CLB CLB BRAM CLB
CLB BRAM CLB CLB BRAM CLB
CLB BRAM CLB CLB BRAM CLB
CLB BRAM CLB CLB BRAM CLB
CLB BRAM CLB CLB BRAM CLB

Routing

On-chip memory

Logic blocks

- Reprogrammable, load arbitrary circuits onto the FPGA
- Once programmed acts similar to an integrated circuit (lower frequency)
- Logic blocks (around 100,000)
- Fast on-chip memory (36K each)

Parameterizable Regular Expression Engine

Regular Expression in Hardware

- Regex can be mapped to a Non-deterministic finite automata (NFA)
- NFAs can be efficiently executed on FPGAs [4,5]

Regular expression: `(ab+|ba+)c`

Input:

[4] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs*, FCCM'01

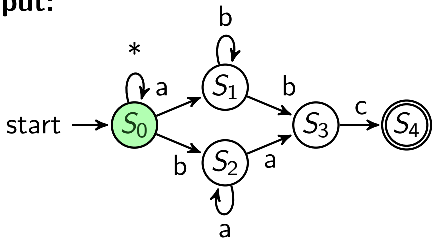
[5] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

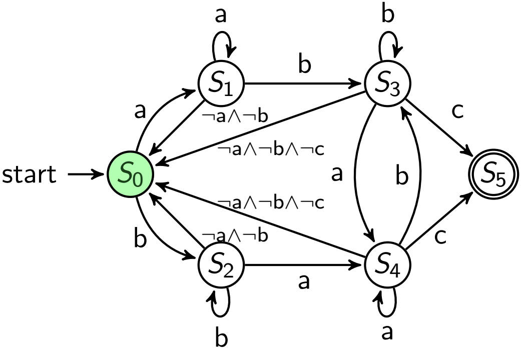
Regular Expression in Hardware

- Regex can be mapped to a Non-deterministic finite automata (NFA)
- NFAs can be efficiently executed on FPGAs [4,5]

Regular expression: `(ab+|ba+)c`

Input:





[4] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs*, FCCM'01

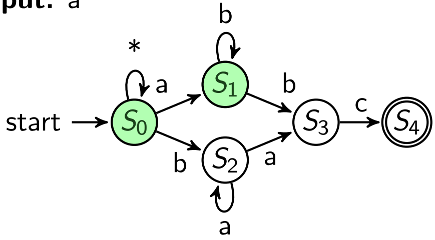
[5] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

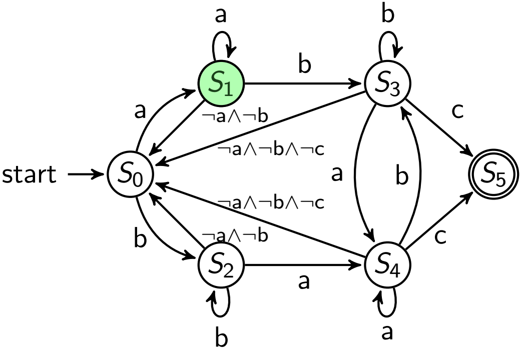
Regular Expression in Hardware

- Regex can be mapped to a Non-deterministic finite automata (NFA)
- NFAs can be efficiently executed on FPGAs [4,5]

Regular expression: `(ab+|ba+)c`

Input: a





[4] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs*, FCCM'01

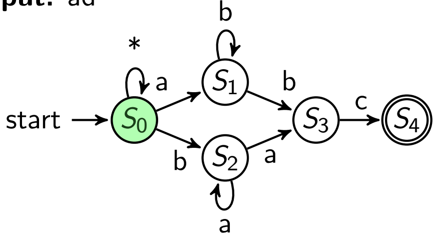
[5] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

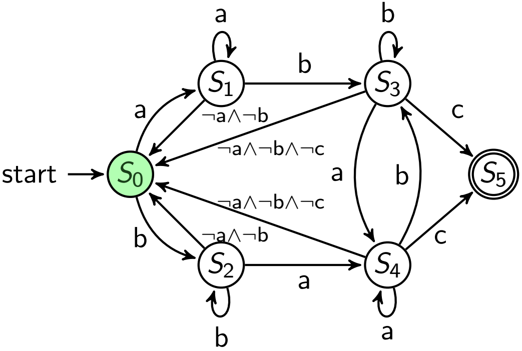
Regular Expression in Hardware

- Regex can be mapped to a Non-deterministic finite automata (NFA)
- NFAs can be efficiently executed on FPGAs [4,5]

Regular expression: `(ab+|ba+)c`

Input: ad





[4] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs*, FCCM'01

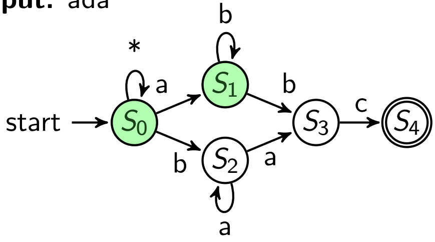
[5] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

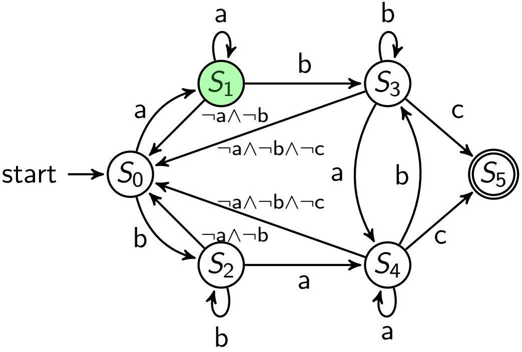
Regular Expression in Hardware

- Regex can be mapped to a Non-deterministic finite automata (NFA)
- NFAs can be efficiently executed on FPGAs [4,5]

Regular expression: `(ab+|ba+)c`

Input: ada





[4] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs*, FCCM'01

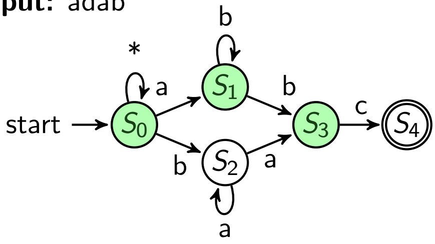
[5] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

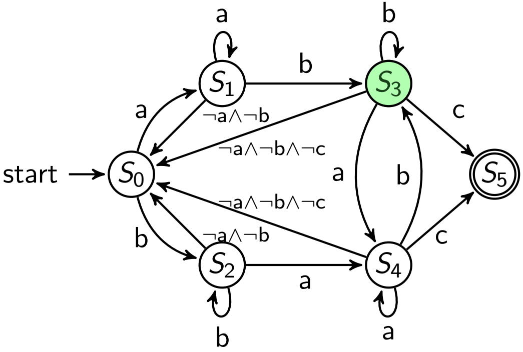
Regular Expression in Hardware

- Regex can be mapped to a Non-deterministic finite automata (NFA)
- NFAs can be efficiently executed on FPGAs [4,5]

Regular expression: `(ab+|ba+)c`

Input: adab





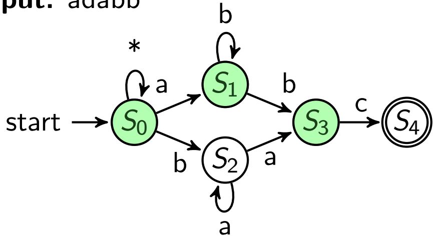
[4] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs*, FCCM'01
[5] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

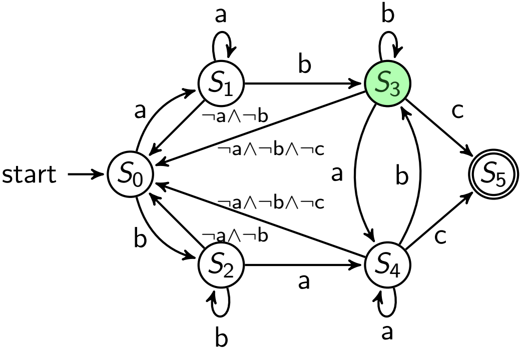
Regular Expression in Hardware

- Regex can be mapped to a Non-deterministic finite automata (NFA)
- NFAs can be efficiently executed on FPGAs [4,5]

Regular expression: `(ab+|ba+)c`

Input: adabb





[4] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs*, FCCM'01

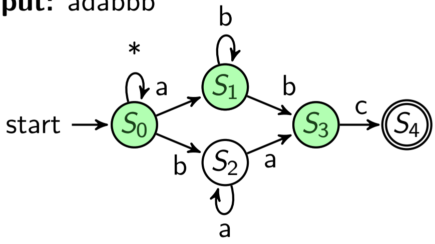
[5] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

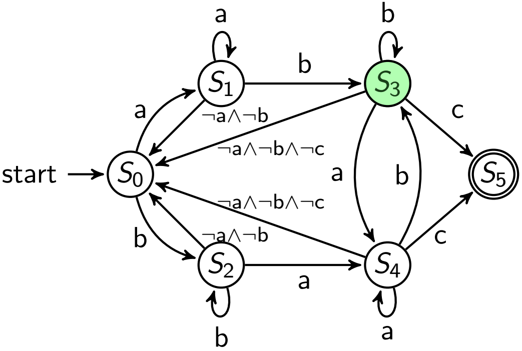
Regular Expression in Hardware

- Regex can be mapped to a Non-deterministic finite automata (NFA)
- NFAs can be efficiently executed on FPGAs [4,5]

Regular expression: `(ab+|ba+)c`

Input: adabbb





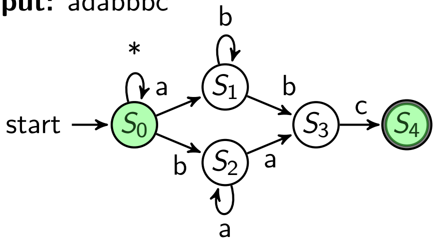
[4] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs*, FCCM'01
[5] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

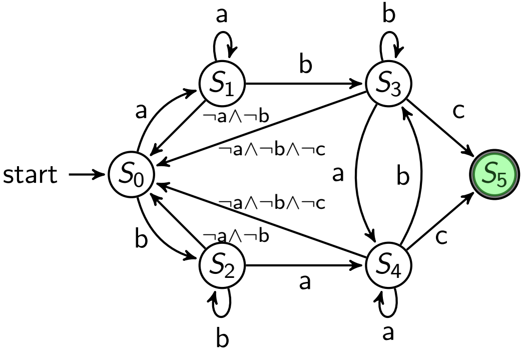
Regular Expression in Hardware

- Regex can be mapped to a Non-deterministic finite automata (NFA)
- NFAs can be efficiently executed on FPGAs [4,5]

Regular expression: `(ab+|ba+)c`

Input: adabbbc



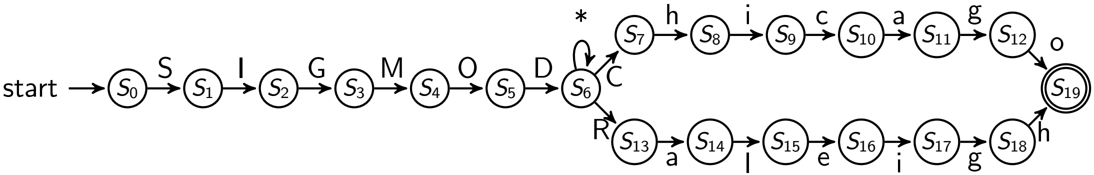


[4] R. Sidhu, V. Prasanna, *Fast regular expression matching using FPGAs*, FCCM'01

[5] L. Woods, J. Teubner, *Complex event detection at wire speed with FPGAs*, VLDB'10

Complexity vs Hardware resources

Regular expression: `SIGMOD.*(Chicago|Raleigh)`

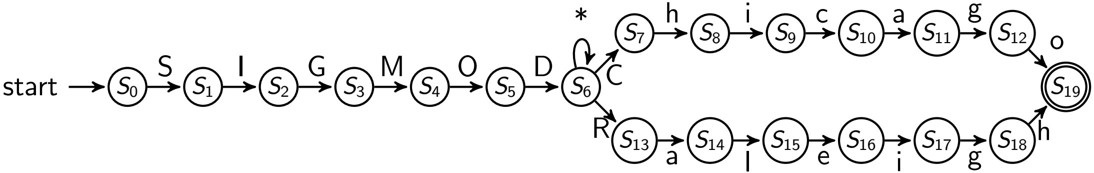


- Resource usage and routing are a crucial factors in FPGA development
- FPGA resource usage grows with regular expression complexity
- If the NFA becomes too large routing/connecting its resources might not be possible

⇒ **Compress the NFA**

NFA compression

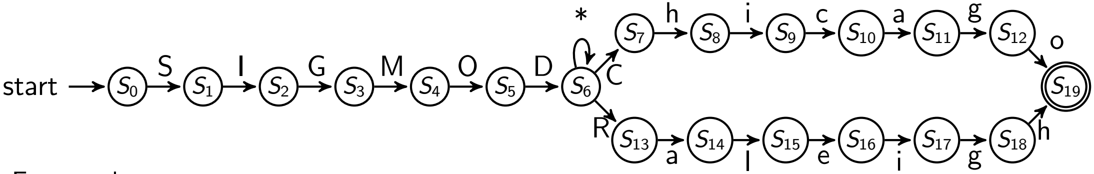
Regular expression: `SIGMOD.*(Chicago|Raleigh)`



[6] J. Teubner, L. Woods, *Skeleton automata for FPGAs: reconfiguring without reconstructing*, SIGMOD'12

NFA compression

Regular expression: `SIGMOD.*(Chicago|Raleigh)`



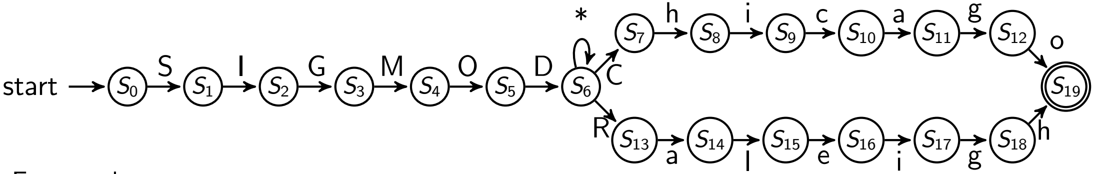
Extracted sequences:

- SIGMOD
- Chicago
- Raleigh

[6] J. Teubner, L. Woods, *Skeleton automata for FPGAs: reconfiguring without reconstructing*, SIGMOD'12

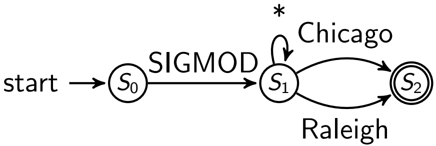
NFA compression

Regular expression: `SIGMOD.*(Chicago|Raleigh)`



Extracted sequences:

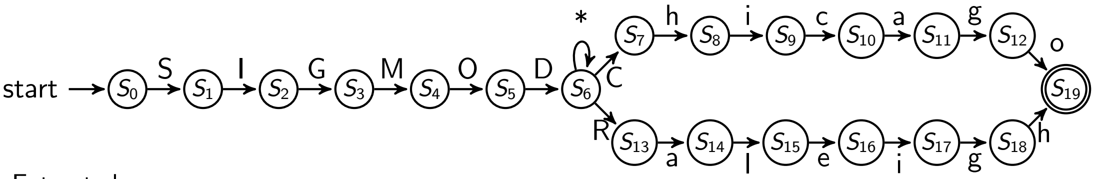
- SIGMOD
- Chicago
- Raleigh



[6] J. Teubner, L. Woods, *Skeleton automata for FPGAs: reconfiguring without reconstructing*, SIGMOD'12

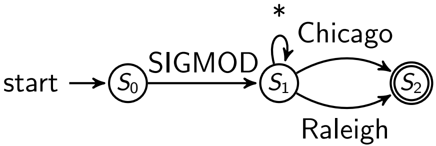
NFA compression

Regular expression: `SIGMOD.*(Chicago|Raleigh)`



Extracted sequences:

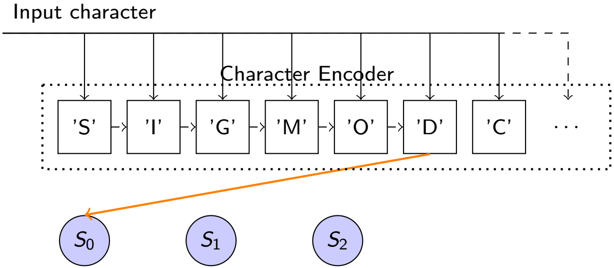
- SIGMOD
- Chicago
- Raleigh



- Decouple character encoding from state transitions in NFA [6]

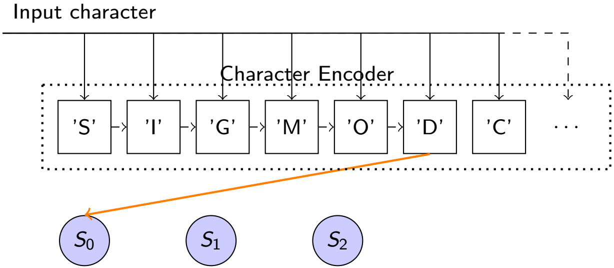
[6] J. Teubner, L. Woods, *Skeleton automata for FPGAs: reconfiguring without reconstructing*, SIGMOD’12

Character Encoder



- Enables compression of NFA by chaining characters into sequences
- Can check for ranges by comparing upper and lower value
- Can support case-insensitivity or collations (e.g., a, ae, ä)

Character Encoder



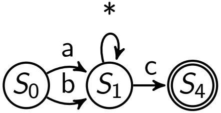
- Enables compression of NFA by chaining characters into sequences
- Can check for ranges by comparing upper and lower value
- Can support case-insensitivity or collations (e.g., a, ae, ä)

Character Encoder can be parametrized at runtime.

Runtime parametrization

Regular expression:

(a|b).*c

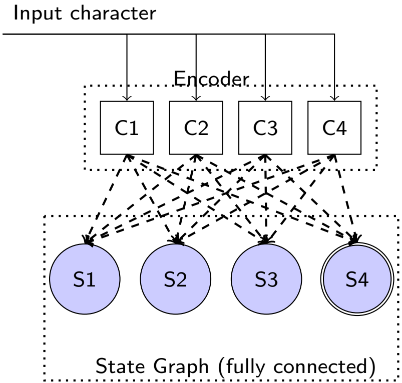


Characters

	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	0	0



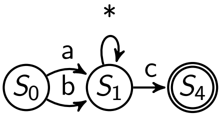
State Transitions

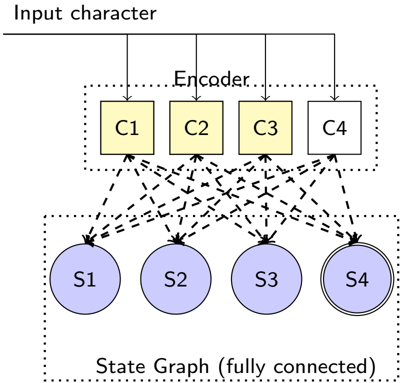
	S1	S2	S3	S4
S1	1	0	0	0
S2	0	0	0	0
S3	0	0	0	0
S4	1	0	0	0

Runtime parametrization

Regular expression:

`(a|b).*c`





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	0	0

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

Runtime parametrization

Characters

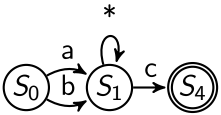
	C1	C2	C3	C4
	'a'	'b'	'c'	

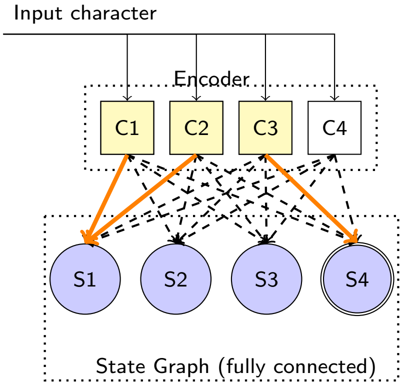
Triggers

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

Regular expression:

`(a|b).*c`





Input character

Encoder

C1 C2 C3 C4

S1 S2 S3 S4

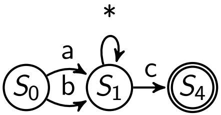
State Graph (fully connected)

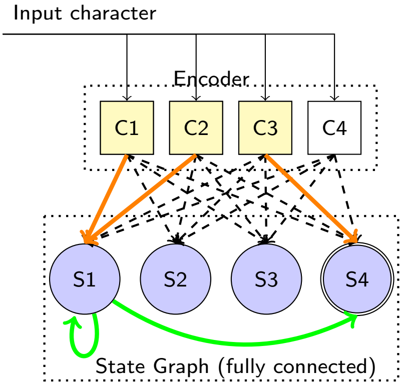
State Transitions

	S1	S2	S3	S4
S1	1	0	0	0
S2	0	0	0	0
S3	0	0	0	0
S4	1	0	0	0

Runtime parametrization

Regular expression:
(a|b).*c





Characters

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

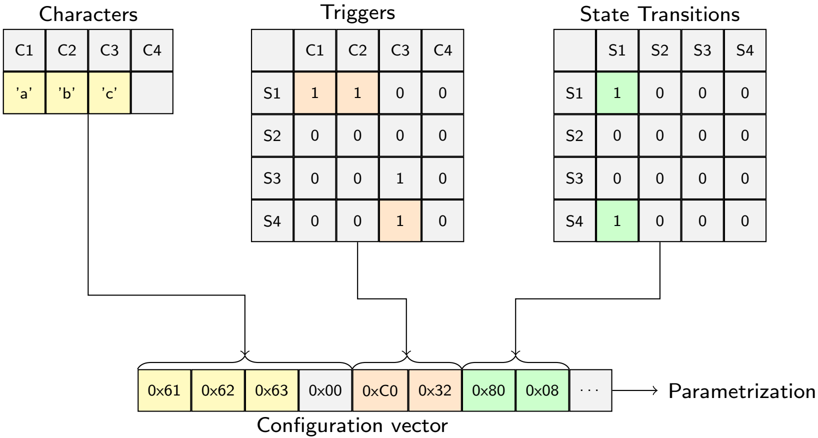
Triggers

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

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

Configuration vector



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

State Transitions

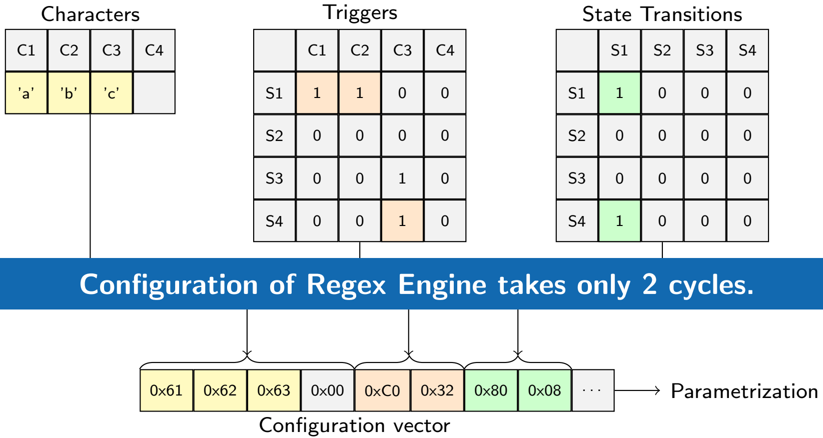
	S1	S2	S3	S4
S1	1	0	0	0
S2	0	0	0	0
S3	0	0	0	0
S4	1	0	0	0

0x61	0x62	0x63	0x00	0xC0	0x32	0x80	0x08	···

Configuration vector

→ Parametrization

Configuration vector



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

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

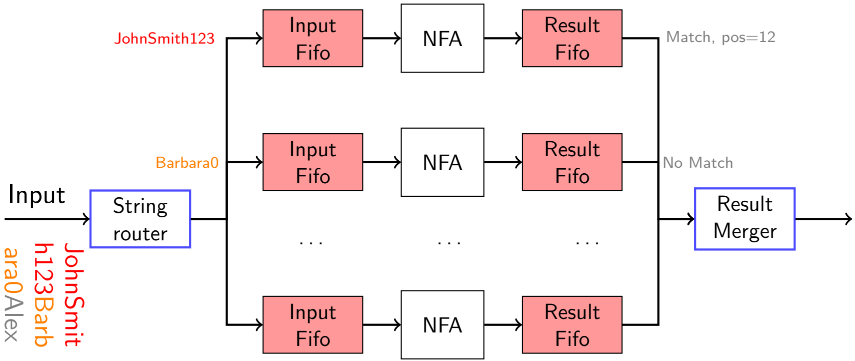
Configuration of Regex Engine takes only 2 cycles.

0x61	0x62	0x63	0x00	0xC0	0x32	0x80	0x08	···

→ Parametrization

Configuration vector

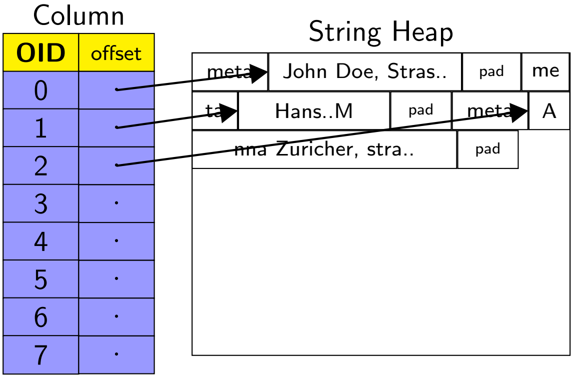
Assembly of a Regex Engine



Integration into Database

Integration into MonetDB

- Column store
- Simple data layout
- Minimize memory bandwidth overhead
- UDF can operate on columns
- Strings are stored in a heap



Column

OID	offset
0	·
1	·
2	·
3	·
4	·
5	·
6	·
7	·

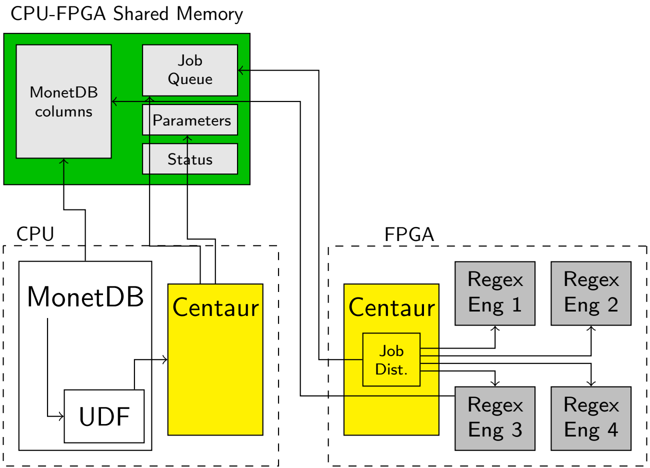
String Heap

meta | John Doe, Stras.. | pad | me

ta | Hans..M | pad | meta | A

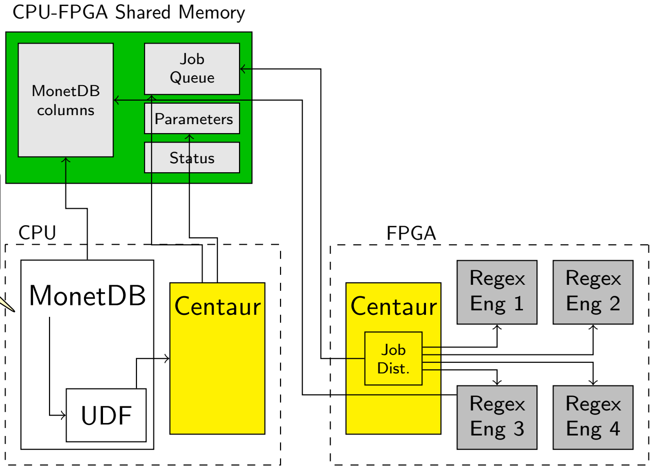
nna Zuricher, stra.. | pad

System Overview



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

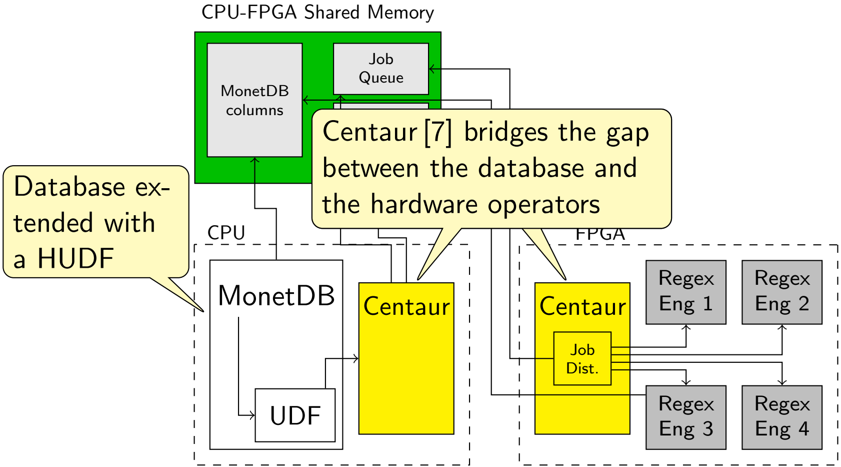
System Overview



Database extended with a HUDF

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

System Overview

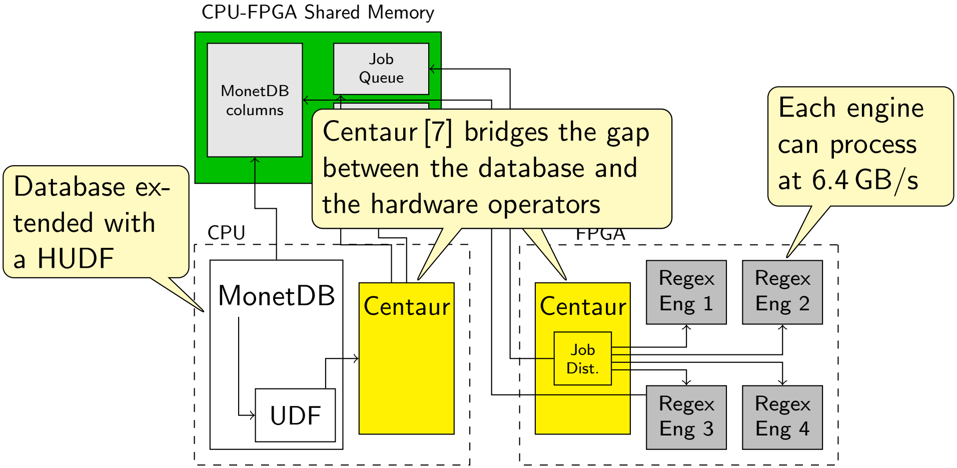


Database extended with a HUDF

Centaur [7] bridges the gap between the database and the hardware operators

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

System Overview



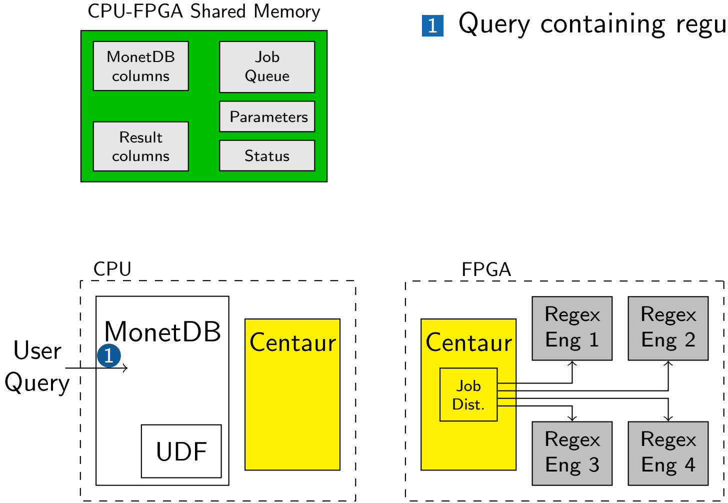
Database extended with a HUDF

Centaur [7] bridges the gap between the database and the hardware operators

Each engine can process at 6.4 GB/s

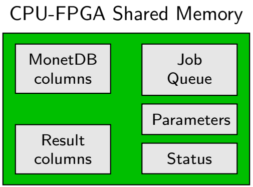
[7] M. Owaida, D. Sidler, *Centaur: A Framework for Hybrid CPU-FPGA Databases*, FCCM'17

Execution Walkthrough

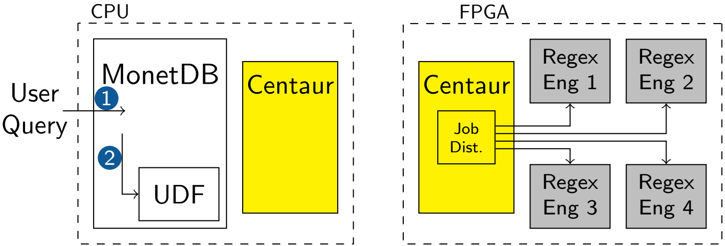


1 Query containing regular expression is submitted

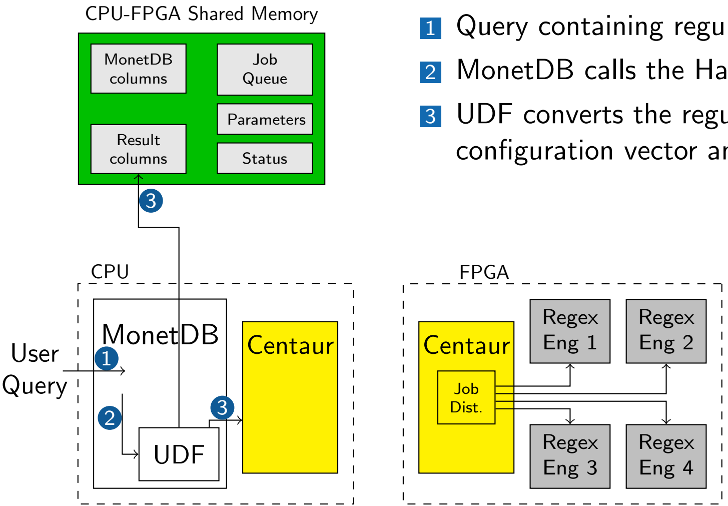
Execution Walkthrough



1. Query containing regular expression is submitted
2. MonetDB calls the Hardware UDF

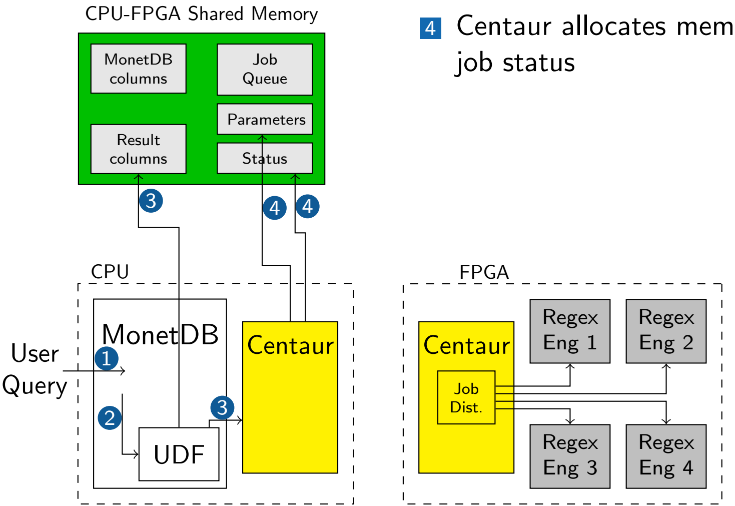


Execution Walkthrough



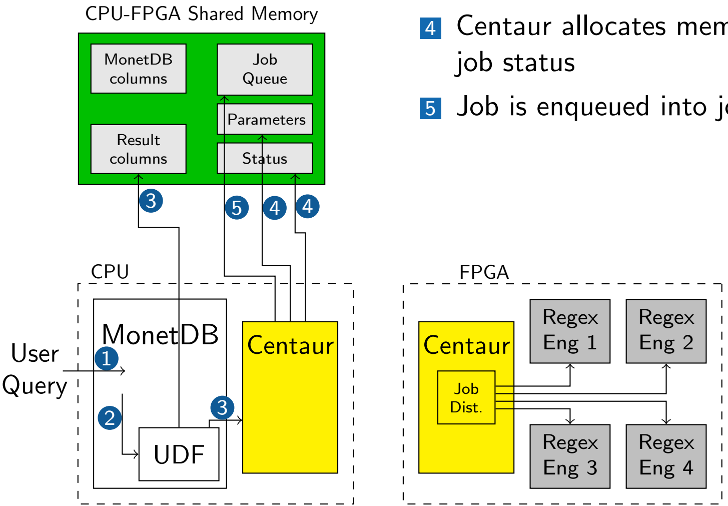
1. Query containing regular expression is submitted
2. MonetDB calls the Hardware UDF
3. UDF converts the regular expression into a configuration vector and allocates the result column

Execution Walkthrough



4 Centaur allocates memory for the job parameters and job status

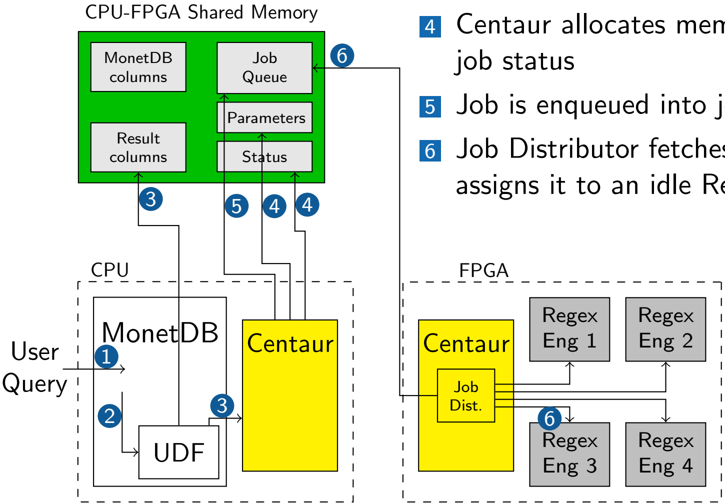
Execution Walkthrough



4 Centaur allocates memory for the job parameters and job status

5 Job is enqueued into job queue

Execution Walkthrough

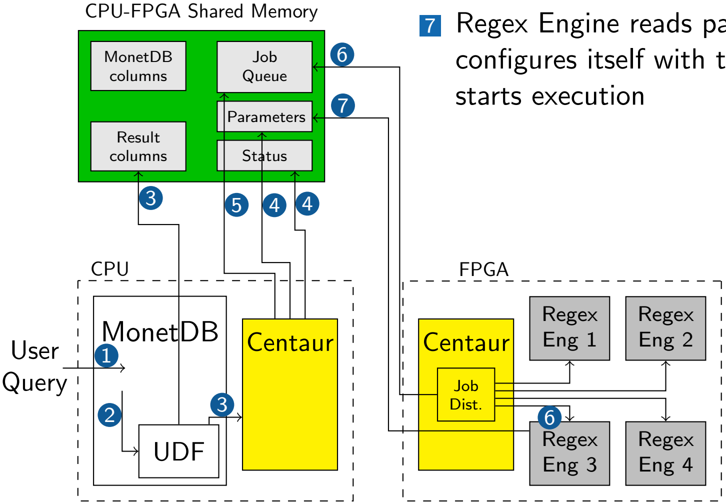


4 Centaur allocates memory for the job parameters and job status

5 Job is enqueued into job queue

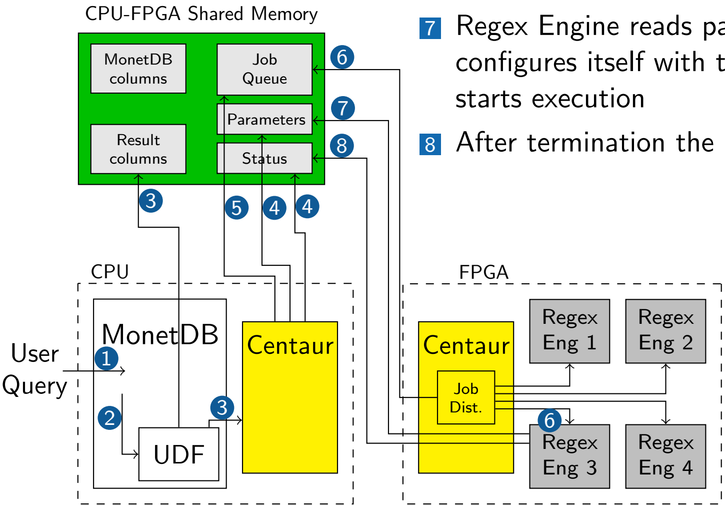
6 Job Distributor fetches the job from the job queue and assigns it to an idle Regex Engine

Execution Walkthrough



7 Regex Engine reads parameters from shared memory, configures itself with the configuration vector and starts execution

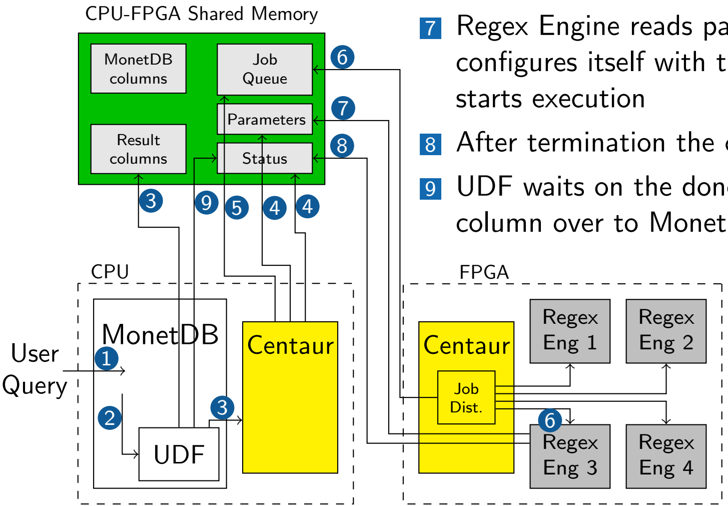
Execution Walkthrough



7 Regex Engine reads parameters from shared memory, configures itself with the configuration vector and starts execution

8 After termination the done bit is set

Execution Walkthrough



7 Regex Engine reads parameters from shared memory, configures itself with the configuration vector and starts execution

8 After termination the `done` bit is set

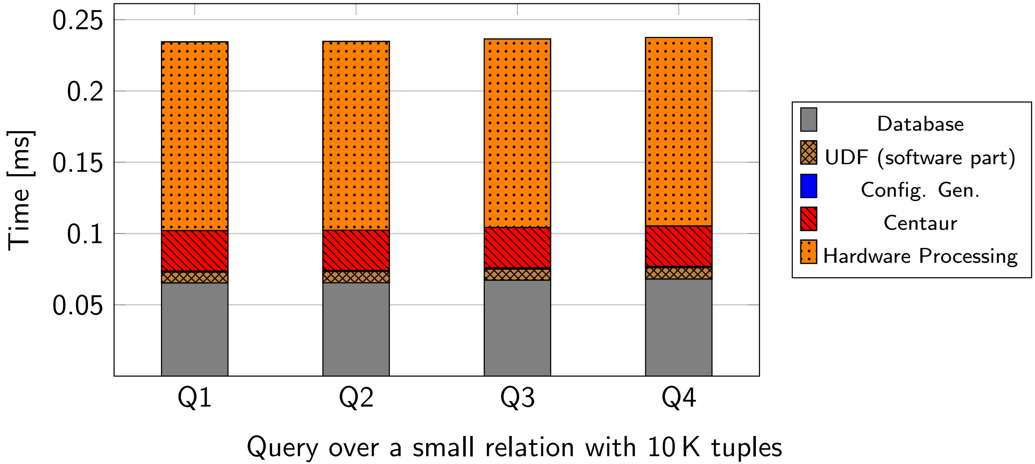
9 UDF waits on the `done` bit and then hands the result column over to MonetDB

Evaluation

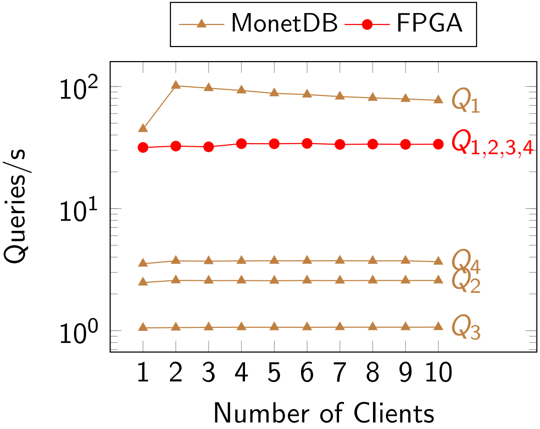
Evaluation - Queries

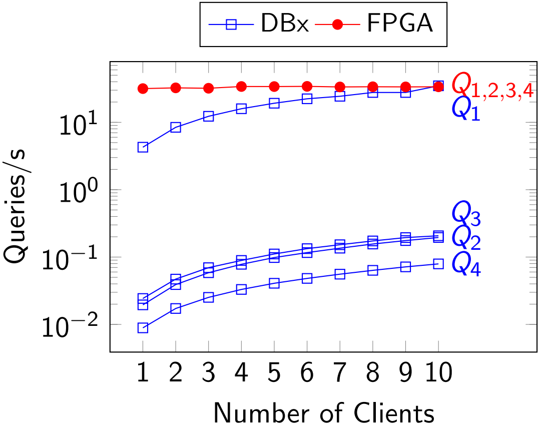
```
Q1:   SELECT count(*) FROM address_table
      WHERE address_string LIKE '%Strasse%';

Q2:   SELECT count(*) FROM address_table
      WHERE REGEXP_LIKE(address_string, '(Strasse|Str\.).*(8[0-9]{4})';

Q3:   SELECT count(*) FROM address_table
      WHERE REGEXP_LIKE(address_string, '[0-9]+(USD|EUR|GBP)');

Q4:   SELECT count(*) FROM address_table
      WHERE REGEXP_LIKE(address_string, '[A-Za-z]{3}\:[0-9]{4}');
```

Evaluation - Microbenchmark

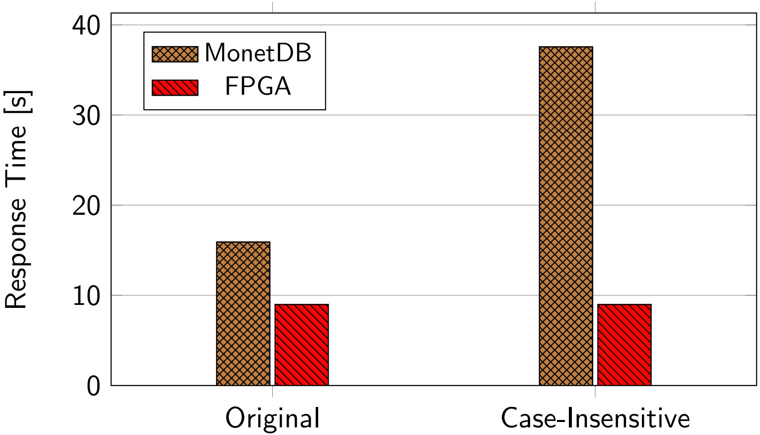


Evaluation - Throughput





Evaluation - TPC-H Q13



Scaling factor set to 0.1 due to limited memory space

Comparison to Accelerators

	GPU [1]	GPU [2]	Xeon Phi [3]	Our work
Regex evaluation	No	Yes	Yes	Yes
Complexity indp. perf.	Yes	No	No	Yes
TP - local data [GB/s]	60-70	10-15	30-40	25.6*
TP - host data [GB/s]	–	1-5	–	6.4
Architecture	fast GDDR memory	fast GDDR memory	60-70 cores, GDDR5 memory	**specialized** core, direct memory access

* Without the memory bandwidth limitation

[1] E. Sitaridi, K. Ross, *GPU-Accelerated string matching for database applications*, VLDB Journal, Oct. 2016
[2] C.-H. Lin, et al., *Accelerating regular expression matching using hierarchical parallel machines on GPU*, GLOBECOM'11
[3] E. Sitaridi, O. Polychroniou, K. Ross, *SIMD-Accelerated regular expression matching*, DAMON'16

Comparison to Accelerators

	GPU [1]	GPU [2]	Xeon Phi [3]	Our work
Regex evaluation	No	Yes	Yes	Yes
Complexity indp. perf.	Yes	No	No	Yes
TP - local data [GB/s]	60-70	10-15	30-40	25.6*
TP - host data [GB/s]	–	1-5	–	6.4
Architecture	fast GDDR memory	fast GDDR memory	60-70 cores, GDDR5 memory	**specialized** core, direct memory access

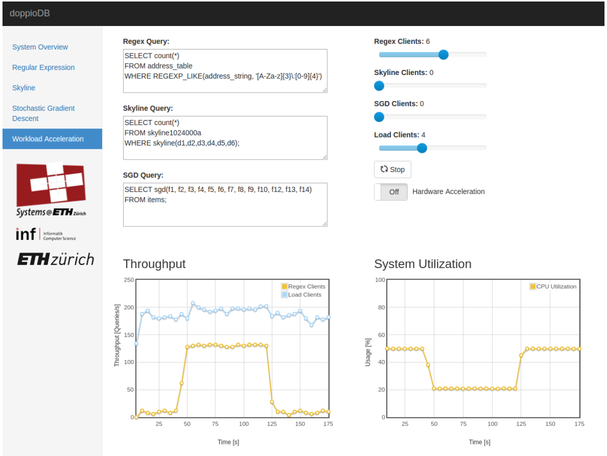
* Without the memory bandwidth limitation

Your next CPU might come with an FPGA!

[2] C.-H. Lin, et al., *Accelerating regular expression matching using hierarchical parallel machines on GPU*, GLOBECOM'11

[3] E. Sitaridi, O. Polychroniou, K. Ross, *SIMD-Accelerated regular expression matching*, DAMON'16

Visit our Demo!



Regex Query:

```
SELECT count(*)
FROM address_table
WHERE REGEXP_LIKE(address_string, '[A-Za-z]{3}\:[0-9]{4}')
```

Skyline Query:

```
SELECT count(*)
FROM skyline1024000a
WHERE skyline(d1,d2,d3,d4,d5,d6);
```

SGD Query:

```
SELECT sgd(f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f12, f13, f14)
FROM items;
```

Regex Clients: 6

Skyline Clients: 0

SGD Clients: 0

Load Clients: 4

Stop

Off Hardware Acceleration

More Information:
systems.ethz.ch/fpga/db_acceleration

Code on GitHub:
github.com/fpgasystems/dobbiodb