doppioDB: A Hardware Accelerated Database

David Sidler   Muhsen Owaida   Zsolt István   Kaan Kara   Gustavo Alonso
Systems Group, Department of Computer Science, ETH Zürich
{firstname.lastname}@inf.ethz.ch

***Abstract*—Relational databases provide a wealth of functionality to a wide range of applications. Yet, there are tasks for which they are less than optimal, for instance when processing becomes more complex (e.g., regular expression evaluation, data analytics) or the data is less structured (e.g., text or long strings). With the increasing amount of user-generated data stored in relational databases, there is a growing need to analyze unstructured text data. At the same time more complex analytical operators are required to extract useful information from the vast amount of collected data. However, many analytical operators incur a significant compute complexity not suitable to database engines where multiple queries share the available resources.**

In this demonstration we show the benefit of using specialized hardware for such tasks and highlight the importance of a flexible, reusable mechanism for extending database engines with hardware-based operators. Our hybrid database engine, *doppioDB*, is deployed on an emerging Xeon+FPGA multicore architecture where the CPU and FPGA have cache-coherent access to the same memory, such that the hardware operators can directly access the database tables. The demonstration is illustrating the acceleration benefits of hardware operators, as well as *doppioDB*'s flexibility in accommodating changing workloads.

I. System Description

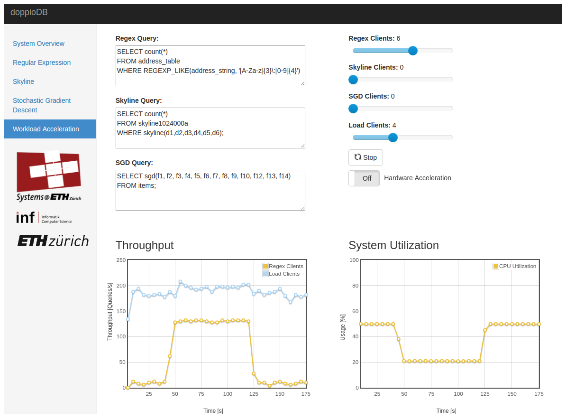
DoppioDB is built on top of MonetDB, a column oriented relational storage engine. Its functionality has been extended with hardware-based operators using the user-defined function (UDF) interface of the database. This allows the seamless integration of hardware operators into any kind of SQL query. For the integration of the operators into MonetDB, we use Centaur [2], a framework that provides a software interface to execute and monitor jobs on the FPGA. Since we are using the Intel Xeon+FPGA platform with shared CPU-FPGA memory, in Centaur all communication between the software and the FPGA occurs through shared memory data structures.

We chose three compute and one data movement intensive operator to showcase the benefits and capabilities of *doppioDB*. First, we show how a regular expression matching operator [3] can speed up query execution. While in software the evaluation of regular expressions can become very costly, especially when they contain wildcards or choices, on FPGAs they can be executed efficiently as non-deterministic finite state automatons (NFAs). Further, our NFA implementation, as we show in this demo, can be quickly re-parametrized at runtime.

The second and third operators we showcase in this demo are analytical ones: stochastic gradient descent (SGD) [1] and skyline [4]. They both are building blocks to more complex machine learning and analytical workloads and as a result an important addition to the database. These operators benefit from the massive parallelism of the FPGA and provide significant speedup over their software counterparts.

Finally, the conditional sum operator evaluates a constant-based comparison on numerical values to determine the sum of matching items. This operator is an example of fusing separate SQL operations into a single hardware operator. Even tough this operator in itself would not warrant FPGA acceleration, we show that it can be pipelined with one of the other operators to reduce data movement between the FPGA and main memory and thereby improving overall system performance.

II. Demo Description



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

Off Hardware Acceleration

Fig. 1: Mixed workload acceleration, hardware acceleration was enabled between timestamp 40 s and 120 s

During the demonstration, the user can interact through a web interface (Figure 1) with the database. The interface consists of multiple tabs, one for each operator, one to illustrate the pipelining of operators, and one to illustrate a mixed workload where multiple clients execute queries concurrently. In the workload demo, queries are continuously executed and hardware acceleration can be enabled/disabled at runtime. As a result the effect of hardware acceleration can be immediately observed by the visitor through an increased throughput and a lower CPU utilization, as visualized in Figure 1. In the other tabs the user can execute single queries on the system either with hardware acceleration enabled or disabled. The user will see the different type of queries *doppioDB* can handle and observe the effect of hardware acceleration through the reported response time.

References

[1] K. Kara, D. Alistarh, C. Zhang, et al. FPGA accelerated dense linear machine learning: A precision-convergence trade-off. In *FCCM'17*.

[2] M. Owaida, D. Sidler, and G. Alonso. Centaur: A framework for hybrid CPU-FPGA databases. In *FCCM'17*.

[3] D. Sidler, Z. István, M. Owaida, and G. Alonso. Accelerating pattern matching queries in hybrid CPU-FPGA architectures. In *SIGMOD'17*.

[4] L. Woods, G. Alonso, and J. Teubner. Parallel computation of skyline queries. In *FCCM'13*.