ReProVide: Query Optimisation and Near-Data Processing on Reconfigurable SoCs for Big Data Analysis [2]

Demo Paper

Tobias Hahn [1], Maximilian Langohr [1], Stefan Meißner [1], Benedikt Döring [1], Stefan Wildermann [1], Klaus Meyer-Wegener [1], and Jürgen Teich [1]

Abstract:

The goal of ReProVide is to provide novel hardware and optimisation techniques for scalable, high-performance processing of Big Data. The Programmable System-on-Chip (PSoC) architecture of ReProVide includes a reconfigurable FPGA for the support of hardware accelerators for various operators on relational and streaming data. Such PSoCs can be used to process data directly at the source, such as data from attached NVMes, using application-specific accelerators. For example, compute-intensive tasks such as JSON parsing can be offloaded to the hardware accelerators, reducing CPU load. In addition, reducing the volume of data at an early stage avoids unnecessary data movements, resulting in lower energy consumption. This demo illustrates the opportunities and benefits of hardware-reconfigurable, FPGA-based PSoCs for near-data processing. The demo allows users to run two queries and select which operations should be pushed onto the SoC for near-data hardware acceleration. From no acceleration to maximum acceleration, a 52× improvement in throughput and 67× lower energy consumption can be observed.

Keywords: Demo, Near-Data Processing, FPGA, Stream Processing

1 Introduction

The exponential growth in the volume, velocity, and variety of data stored on servers around the world presents significant challenges. Efficiently analyzing petabytes of data within a reasonable amount of time and energy budget requires large-scale parallel data processing at the source. As a remedy, current research proposes new hardware architectures to reduce data volume early in the processing chain. To take advantage of these novel systems, new query analysis and optimisation techniques are needed.

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany,
tobias.hahn@fau.de, https://orcid.org/0009-0005-4513-0161;
maximilian.langohr@fau.de, https://orcid.org/0000-0002-2520-7134;
stefan.meissner@fau.de, https://orcid.org/0009-0002-8722-1457;
benedikt.lw.doering@fau.de, https://orcid.org/0009-0008-0406-7700;
stefan.wildermann@fau.de, https://orcid.org/0000-0002-4324-2187;
klaus.meyer-wegener@fau.de, https://orcid.org/0000-0002-8102-1019;
juergen.teich@fau.de, https://orcid.org/0000-0001-6285-5862

[2] This work has been supported by the German Science Foundation (Deutsche Forschungsgemeinschaft, DFG) as part of the Priority Programme SPP 2037 under project ReProVide (project number 361498444).

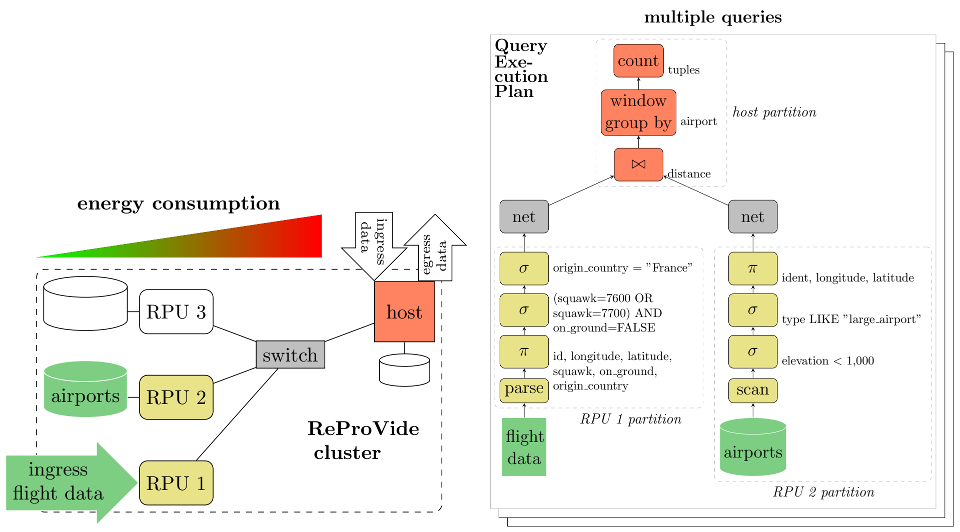


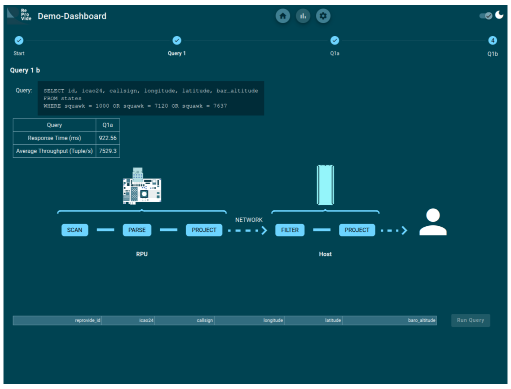
Fig. 1: Overview of a ReProVide cluster (left) where multiple RPUs are connected to a host, which schedules multiple applications (as shown by example right) on the cluster. RPUs can process relational data originating from local storage devices as well as ingress streaming data from external sources.

ReProVide (Reconfigurable Data Provider), as depicted in Fig. 1, proposes and explores FPGA-based solutions for intelligent storage and near-data processing, coupled with new query optimisation methods. Our approach exploits the speed and flexibility of FPGA technology to provide scalable, efficient pre-filtering of Big Data.

The ReProVide demo features a cluster of FPGA-based Programmable System-on-Chip (PSoC) architectures called Reconfigurable Data-Provider Units (RPUs) (see Fig. 2a). RPUs can function as storage-attached devices, interfacing directly with two NVMe SSDs, as well as network-attached devices, processing incoming streaming data from a 10Gbit Ethernet interface. For processing and filtering, RPUs take advantage of the dynamic, run-time reconfigurability of modern FPGAs to load pre-designed hardware accelerators on demand. ReProVide enables hardware-based processing of user-defined queries. The query-specific filtering performed by RPUs significantly reduces the vast amount of data at the source, thereby minimising one of the primary drivers of energy consumption in data center networks: data movement [Bo18].

To integrate RPUs into a Database Management System (DBMS) and take full advantage of their capabilities, we use novel optimisation techniques to optimise for multiple objectives (e.g., latency, throughput). These techniques determine which operations are best suited for execution on RPUs (see Fig. 1, right), utilising cost models that account for the performance and characteristics of RPUs. Additionally, the optimiser decides how to deploy and execute the assigned sub-queries or database operators on hardware accelerators, which are mapped to RPUs via hardware reconfiguration.

(a) ReProVide cluster



Query 1 b

Query:

```
SELECT id, icao24, callsign, longitude, latitude, bar_altitude
FROM states
WHERE squawk = 1000 OR squawk = 7120 OR squawk = 7637
```

Query	Q1a
Response Time (ms)	922.56
Average Throughput (Tuple/s)	7529.3

(b) Interactive demo dashboard

Fig. 2: In the ReProVide demo, two queries using different RPU accelerators can be executed live on the ReProVide cluster (a). The query results and statistics can be viewed directly on the dashboard (b).

2 Related Work

FPGAs are emerging as a promising architecture for many Big Data applications. This is due to their ability to implement parallel, deeply pipelined hardware accelerators that are perfectly tailored to specific operations. In addition, FPGAs support run-time reconfiguration, allowing hardware accelerators to be dynamically swapped out to adapt to changing queries. FPGAs can be attached directly to CPUs as *co-processors*, as proposed in [CO14; KT11; Ma19; Wa16; Zi16]. The CPU is responsible for transferring the data to the local memory of the FPGA, which can then process this input [Fa20]. This results in a lot of additional data movement, making this approach less energy efficient [Bo18].

In *shared memory systems*, such as those proposed in [Mo23; Si17a; Si17b; St15], the CPU and FPGA can both access the same main memory [Fa20]. While this helps to avoid additional memory transfers, the CPU and FPGA share not only the same memory but also the memory bus. The bandwidth required by the accelerator can therefore limit the processing speed of the CPU. *Near-data processing* systems such as [Be15; Be22; MTA10; TWN13; WIA14] differ significantly. Here, the FPGA is placed between the data source and the CPU. Even then it might not be possible to fully process a query on the FPGA, but it can significantly accelerate certain tasks such as filtering data. ReProVide follows the near-data processing design principle.

3 Demonstration Setup

The ReProVide demo shows a ReProVide cluster consisting of 2 RPUs (see Fig. 2a) and an Intel(R) Core(TM) i9-13900K host computer. The demo also includes a monitor connected to the host that displays the ReProVide dashboard (see Fig. 2b). The dashboard can be used to launch and monitor queries live on the cluster. The RPUs and the host are connected via a 10 GBit network. In addition, a 1 GBit network is used, through which the host is able to control and reconfigure the RPUs.

Xilinx Zynq ZCU106 PSoCs are used for our RPUs. These consist of a programmable logic and a processing system, which includes a quad-core ARM® Cortex®-A53 applications processor. The programmable logic is divided into a static region and four reconfigurable regions. Depending on the application, the associated accelerators are dynamically loaded onto these reconfigurable regions. The static region contains DMA engines, IP cores for NVMe and Ethernet interfacing, and interconnection logic. All data is moved between accelerators and interfaces using DMAs, managed by the ARM CPU.

4 Walkthrough

The ReProVide demo shows the execution of two selected queries from the SKYSHARK benchmark [LVM23], which was specially developed for testing and evaluating hardware-accelerated database systems. The SKYSHARK benchmark demonstrates various flight monitoring analysis applications using real flight tracking data collected by OpenSky Network. The following two queries were selected from the benchmark for the ReProVide demo:

```
SELECT id,icao24,callsign,longitude,latitude,baro_altitude
FROM states
WHERE squawk = 1000 OR squawk = 7120 OR squawk = 7637
```

Query 1: Searches for flight data tuples with specific transponder codes (squawk).

```
SELECT id,icao24,callsign,longitude,latitude,baro_altitude
FROM states
WHERE baro_altitude > 10668.0 AND
    (vertical_rate < -0.33 OR
    ((squawk = 8600 OR squawk = 1000) AND vertical_rate < 10.15))
```

Query 2: Searches for flight data tuples with a more complex filtering condition based on altitude, vertical velocity, and transponder code.

The QEPs of both queries are a sequence of parse, selection σ, and projection π operators, and consequently follow the form as also depicted in Fig. 3 (left). The ReProVide optimiser creates various execution plans for each query and then selects the plan that best meets the given requirements regarding response time, throughput, and energy consumption. A cost model specifically designed for such FPGA-accelerated systems is used to quickly evaluate and compare different plans. In this demo, however, we want to run several of the possible execution plans one after the other in order to observe their effect in the optimisation objectives. Different plans execute different shares of operators either on the RPU or on the host, as indicated in Fig. 3 Plans A to D.

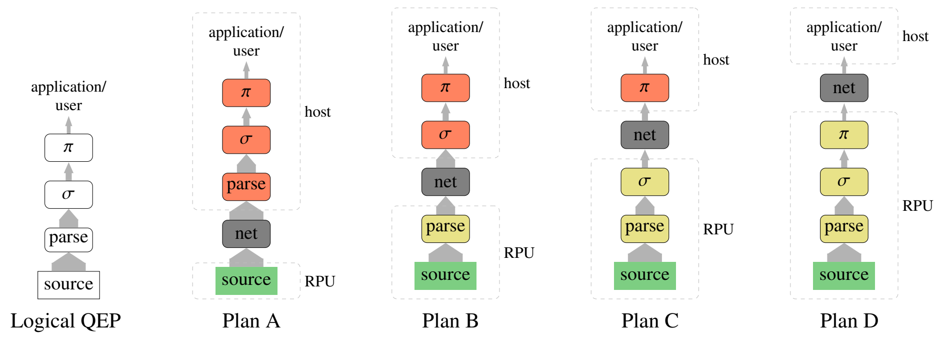


Fig. 3: Example of a logical query execution plan (left) and four possible execution plans for partitioning operators (parsing, selection σ, projection π) between RPU and host.

To execute operators on the RPU, a variety of accelerator cores were developed for ReProVide, focusing on the processing of semi-structured data such as JSON [Ha22; Ha23; Ha24; HWT22; HWT23; HWT24]. These accelerators are perfectly suited for the SKYSHARK benchmark, which also relies on JSON-encoded input data. When the optimiser selects an execution plan, an accelerator is automatically generated and synthesized for all RPU operations in the given plan. Unfortunately, the generation of accelerators at runtime cannot be shown, as this would consume too much time. However, for a Big Data application that runs for hours or even weeks, 10 minutes to synthesize a custom accelerator may be considered reasonable..

In the demo, the four different plans from Fig. 3 are executed one after the other for each of the two queries. For execution plans B, C, and D, the corresponding accelerators were synthesized in advance. No accelerator is required for plan A, as the loaded data on the RPU is transmitted unprocessed. At the start of a demo run (i.e., execution of a query plan), the required accelerator is loaded into a free reconfigurable area on the FPGA using dynamic partial reconfiguration for plans B, C and D. For plan B, the accelerator will parse the incoming JSON data into a C-struct format. In plan C, the accelerator will additionally apply the select operator to filter out unwanted records. Finally, in Plan D, the accelerator also applies the projection operator, reducing the number of attributes and, hence, the total amount of data passed to the host.

After reconfiguration, a 4 GB JSON file is read from an NVMe SSD. The read data is transferred to the accelerator via DMAs, where data is continuously processed. The result data of the accelerator is again moved via DMAs to the 10 GBit interface for further transmission to the host. For execution plan A, the data is transferred directly from the NVMe to the Ethernet interface. The remaining processing steps are then performed on the host (see Fig. 3). Finally, the query result tuples are plotted live on the dashboard and latency and throughput statistics are displayed.

5 Evaluation

When running the demo, the results of each execution plan run are displayed on the dashboard. The evaluation results of a full demo execution are shown in Tab. 1 (not all the information shown is logged in the live demo).

Tab. 1: Evaluation results from the demo execution.

		execution time [s]	response time [s]	throughput [kT/s]	avg. power [W] RPU	avg. power [W] host	avg. power [W] total[†]	total[†] energy [kJ]
Q1	A	400.0	1.17	2954	26	114	162	64.66
	B	12.8	3.25	91944	26	74	122	1.57
	C	9.3	3.29	126979	26	70	119	1.11
	D	8.7	3.17	136153	26	66	114	0.99
Q2	A	400.2	1.14	891	26	114	162	64.67
	B	13.5	3.19	28702	26	76	124	1.68
	C	8.9	3.18	43535	26	66	115	1.02
	D	8.3	3.14	46626	26	67	115	0.96

[†] *total power/energy includes RPU, host and switch.*

The energy measurements were taken using a socket power meter and therefore cover the entire system, including the power supply. It can be observed that the more operators are accelerated on the RPU, the more the throughput increases and the energy consumption decreases. The biggest improvement can be observed between Plan A and Plan B, stemming from the acceleration of compute-intensive JSON parsing. The high CPU utilisation of the host is also reflected in its power consumption. For plan C and D, the CPU utilisation and thus the CPU power consumption is only slightly reduced. However, by reducing the data movements, an additional significant improvement in execution time (i.e., start to last tuple), throughput and energy consumption can be achieved.

The response time (i.e., start to first tuple) shows an increase of about 2 seconds from plan A to the other plans. This increase is due to the additional time required to reconfigure the FPGA with the new accelerator when starting the query.

6 Conclusion

The presented demo elucidates that FPGAs are beneficial for processing data close to the source in Big Data applications. As a result, the CPU is relieved and unnecessary data movements are prevented, positively impacting throughput as well as energy consumption. The freed CPU cycles can then be used to scale across multiple RPUs in order to serve multiple tenants.

References

[Be15] Becher, A.; Ziener, D.; Meyer-Wegener, K.; Teich, J.: A co-design approach for accelerated SQL query processing via FPGA-based data filtering. Pp. 192–195, 2015, DOI: 10.1109/FPT.2015.7393148, URL: https://doi.org/10.1109/FPT.2015.7393148.

[Be22] Becher, A.: Near-Data Query Processing on Heterogeneous FPGA-based Systems, PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2022, URL: https://nbn-resolving.org/urn:nbn:de:bvb:29-opus4-189289.

[Bo18] Boroumand, A.; Ghose, S.; Kim, Y.; Ausavarungnirun, R.; Shiu, E.; Thakur, R.; Kim, D.; Kuusela, A.; Knies, A.; Ranganathan, P.; Mutlu, O.: Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks. In: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2018, Williamsburg, VA, USA, March 24-28, 2018. ACM, pp. 316–331, 2018, DOI: 10.1145/3173162.3173177, URL: https://doi.org/10.1145/3173162.3173177.

[CO14] Casper, J.; Olukotun, K.: Hardware acceleration of database operations. In: The 2014 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '14, Monterey, CA, USA - February 26 - 28, 2014. ACM, pp. 151–160, 2014, URL: https://doi.org/10.1145/2554688.2554787.

[Fa20] Fang, J.; Mulder, Y. T. B.; Hidders, J.; Lee, J.; Hofstee, H. P.: In-memory database acceleration on FPGAs: a survey. VLDB J. 29 (1), pp. 33–59, 2020, DOI: 10.1007/S00778-019-00581-W, URL: https://doi.org/10.1007/s00778-019-00581-w.

[Ha22] Hahn, T.; Becher, A.; Wildermann, S.; Teich, J.: Raw Filtering of JSON Data on FPGAs. In (Bolchini, C.; Verbauwhede, I.; Vatajelu, I., eds.): 2022 Design, Automation & Test in Europe Conference & Exhibition, DATE 2022, Antwerp, Belgium, March 14-23, 2022. IEEE, pp. 250–255, 2022, DOI: 10.23919/DATE54114.2022.9774696, URL: https://doi.org/10.23919/DATE54114.2022.9774696.

[Ha23] Hahn, T.; Schüll, D.; Wildermann, S.; Teich, J.: An FPGA Avro Parser Generator for Accelerated Data Stream Processing. In (König-Ries, B.; Scherzinger, S.; Lehner, W.; Vossen, G., eds.): Datenbanksysteme für Business, Technologie und Web (BTW 2023), 20. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme"(DBIS), 06.-10, März 2023, Dresden, Germany, Proceedings. Vol. P-331. LNI, Gesellschaft für Informatik e.V., pp. 729–749, 2023, DOI: 10.18420/BTW2023-46, URL: https://doi.org/10.18420/BTW2023-46.

[Ha24] Hahn, T.; Schüll, D.; Wildermann, S.; Teich, J.: ABACUS: ASIP-Based Avro Schema-Customizable Parser Acceleration on FPGAs. In: 27th International Symposium on Design & Diagnostics of Electronic Circuits & Systems, DDECS 2024, Kielce, Poland, April 3-5, 2024. IEEE, pp. 79–85, 2024, DOI: 10.1109/DDECS60919.2024.10508904, URL: https://doi.org/10.1109/DDECS60919.2024.10508904.

[HWT22] Hahn, T.; Wildermann, S.; Teich, J.: Auto-Tuning of Raw Filters for FPGAs. In: 32nd International Conference on Field-Programmable Logic and Applications, FPL 2022, Belfast, United Kingdom, August 29 - Sept. 2, 2022. IEEE, pp. 167–175, 2022, DOI: 10.1109/FPL57034.2022.00036, URL: https://doi.org/10.1109/FPL57034.2022.00036.

[HWT23] Hahn, T.; Wildermann, S.; Teich, J.: SPEAR-JSON: Selective Parsing of JSON to Enable Accelerated Stream Processing on FPGAs. In (Mentens, N.; Sousa, L.; Trancoso, P.; Papadopoulou, N.; Sourdis, I., eds.): 33rd International Conference on Field-Programmable Logic and Applications, FPL 2023, Gothenburg, Sweden, September 4-8, 2023. IEEE, pp. 189–196, 2023, DOI: 10.1109/FPL60245.2023.00034, URL: https://doi.org/10.1109/FPL60245.2023.00034.

[HWT24] Hahn, T.; Wildermann, S.; Teich, J.: JSON-CooP: A JSON Decompression/Parsing Co-Design for FPGAs. In: IEEE Proceedings of the 34th International Conference on Field-Programmable Logic and Applications, Turin. 2024, DOI: 10.1109/FPL64840.2024.00012.

[KT11] Koch, D.; Torresen, J.: FPGASort: a high performance sorting architecture exploiting run-time reconfiguration on FPGAs for large problem sorting. In: Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays. FPGA '11, Association for Computing Machinery, Monterey, CA, USA, pp. 45–54, 2011, ISBN: 9781450305549, URL: https://doi.org/10.1145/1950413.1950427.

[LVM23] Langohr, M. S.; Vogler, T.; Meyer-Wegener, K.: SKYSHARK: A Benchmark with Real-world Data for Line-rate Stream Processing with FPGAs. CEUR Workshop Proceedings 3630, pp. 98–109, 2023, URL: https://ceur-ws.org/Vol-3630/LWDA2023-paper9.pdf.

[Ma19] Manev, K.; Vaishnav, A.; Kritikakis, C.; Koch, D.: Scalable Filtering Modules for Database Acceleration on FPGAs. In: 10th Int. Symp. on Highly-Efficient Accelerators and Reconfigurable Technologies, HEART 2019, Nagasaki, Japan, June 6-7, 2019. ACM, 4:1–4:6, 2019, URL: https://doi.org/10.1145/3337801.3337810.

[Mo23] Moghaddamfar, M.: Database System Acceleration on FPGAs, PhD thesis, Technische Universität Dresden, 2023, URL: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-856076.

[MTA10] Müller, R.; Teubner, J.; Alonso, G.: Glacier: a query-to-hardware compiler. In: SIGMOD. ACM, pp. 1159–1162, 2010, URL: https://doi.org/10.1145/1807167.1807307.

[Si17a] Sidler, D.; István, Z.; Owaida, M.; Alonso, G.: Accelerating Pattern Matching Queries in Hybrid CPU-FPGA Architectures. In: SIGMOD. ACM, pp. 403–415, 2017, URL: https://doi.org/10.1145/3035918.3035954.

[Si17b] Sidler, D.; Owaida, M.; István, Z.; Kara, K.; Alonso, G.: doppioDB: A hardware accelerated database. In: 27th Int. Conf. on Field Programmable Logic and Applications, FPL 2017, Ghent, Belgium, Sept. 4-8. IEEE, p. 1, 2017, URL: https://doi.org/10.23919/FPL.2017.8056864.

[St15] Stuecheli, J.; Blaner, B.; Johns, C. R.; Siegel, M. S.: CAPI: a coherent accelerator processor interface. IBM J. Res. Dev. 59 (1), 7:1–7:7, 2015, ISSN: 0018-8646, URL: https://doi.org/10.1147/JRD.2014.2380198.

[TWN13] Teubner, J.; Woods, L.; Nie, C.: XLynx—An FPGA-based XML filter for hybrid XQuery processing. ACM Trans. Database Syst. 38 (4), 2013, ISSN: 0362-5915, URL: https://doi.org/10.1145/2536800.

[Wa16] Wang, Z.; Paul, J.; Cheah, H. Y.; He, B.; Zhang, W.: Relational query processing on OpenCL-based FPGAs. In: 26th Int. Conf. on Field Programmable Logic and Applications, FPL, Lausanne, Switzerland, Aug. 29 - Sept. 2. IEEE, pp. 1–10, 2016, URL: https://doi.org/10.1109/FPL.2016.7577329.

[WIA14] Woods, L.; István, Z.; Alonso, G.: Ibex: an intelligent storage engine with support for advanced SQL offloading. PVLDB 7 (11), pp. 963–974, 2014, ISSN: 2150-8097, URL: https://doi.org/10.14778/2732967.2732972.

[Zi16] Ziener, D.; Bauer, F.; Becher, A.; Dennl, C.; Meyer-Wegener, K.; Schürfeld, U.; Teich, J.; Vogt, J.; Weber, H.: FPGA-Based Dynamically Reconfigurable SQL Query Processing. ACM Trans. Reconfigurable Technol. Syst. 9 (4), 25:1–25:24, 2016, DOI: 10.1145/2845087, URL: https://doi.org/10.1145/2845087.