Leveraging FPGAs to Accelerate the Query Processing of SQL-Based DataBases

Behzad Salami
Computer Science Department
Barcelona Supercomputing Center (BSC)
Barcelona, Spain
Email: behzad.salami@bsc.es

Osman Unsal
Computer Science Department
Barcelona Supercomputing Center (BSC)
Barcelona, Spain
Email: osman.unsal@bsc.es

Adrian Cristal Kestelman
Computer Science Department
Barcelona Supercomputing Center (BSC)
Barcelona, Spain
Email: adrian.cristal@bsc.es

***Abstract*—With the rise of Big Data, providing high-performance query processing capabilities through the acceleration of database analytics has gained significant attention. Leveraging Field Programmable Gate Array (FPGA) technology, this approach can lead to clear benefits. In this work, we briefly introduce the design and implementation of an FPGA-based platform that enables fast query processing for database systems by melding novel database-specific accelerators with commercial-off-the-shelf (COTS) storage using modern interfaces, in a novel, unified, and a programmable environment. The proposed engine can perform a large subset of SQL queries through its set of instructions that can map compute-intensive database operations, such as filter, arithmetic, aggregate, group by, table join, or sort, on to the specialized high-throughput accelerators. To minimize the amount of SSD I/O operations required, it also supports hardware MinMax indexing for databases. We evaluated our query processing engine with five decision support queries from the TPC-H benchmark suite and achieved a speedup from 1.8X to 34.2X, in comparison to the state-of-the-art DBMS, i.e., PostgreSQL and MonetDB.**

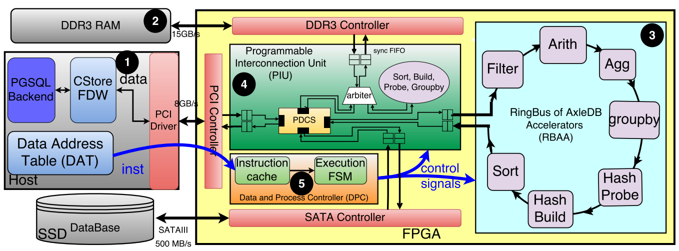


Fig. 1. Overall architecture of the proposed engine with its major components. (1) software extensions for DBMS in the host, (2) Data storage units and device controllers, (3) Query accelerators that are organized in a clockwise unidirectional ring bus (RBAA), (4) Programmable Interconnection Unit (PIU) to manage the accesses to the off-chip data storage units, in a fully flexible fashion, (5) Data and Process Controller (DPC) to orchestrate the involved modules of the engine to process the SQL queries.

I. Introduction

FPGAs provide a unique opportunity to build an efficient query processing platform, by constructing a high-throughput execution engine with the additional aim of minimizing overheads of data movement [1]. It is mainly the consequence of; ***(i)*** the inherent characteristics of massively parallel and configurable architecture of FPGAs, suitable for data streaming in deep pipelined-style execution ***(ii)*** the rise of High-Level Synthesis (HLS) technology, which makes FPGA applications relatively easier to develop compared to low-level languages such as VHDL or Verilog, and ***(iii)*** the availability of soft cores that implement modern interfaces, such as PCIe 3.0 (Peripheral Component Interconnect Express) or SATA-3 (Serial AT Attachment).

For the query processing, FPGAs have been utilized in two distinct approaches: ***(i)*** traditional data offloading mechanisms, where data in the host-attached storage is offloaded towards external processing units or accelerators implemented in the FPGA [2], or ***(ii)*** placing the processing units directly in the data path between the host machine and the main storage units [3]. The first approach could incur overheads stemming from the additional data movement since data needs to be offloaded through the Operating System (OS) and device driver layers. In contrast, the second approach allows processing units to get direct access to the data blocks and facilitates low-latency data transmission. In addition, it can correspond to significant speedups in the query execution. In this work, we follow the second approach to minimize the data transmission overheads.

In a nutshell, the main objectives of this work as a novel query processing engine, are listed as below:

- to provide an infrastructure that sits between the host and the data storage in SSD, and utilizes PCIe-3 and SATA-3 interfaces to work directly with blocks of database columns
- to design a set of efficient query accelerators inside such an infrastructure that can facilitate the query processing in a fully pipelined fashion
- to allow the DBMS to utilize these accelerators by issuing query-specific instructions, exposing flexibility in the data movement and in enabling accelerators

II. The Overall Architecture

In the proposed query processing engine, the host communicates with the FPGA through an Application Program Interface (API), to transfer data and instructions using the PCIe-3 interface. When the host initiates the query execution, the query plan needs to be converted into our specific instructions. Inside the FPGA, these instructions are managed and executed by the Data and Process Controller (DPC), which orchestrates the movement of data blocks between the SSD, DDR-3, hist, and

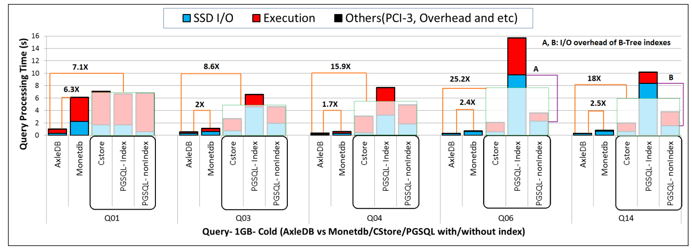
Accelerators. The query is effectively executed by streaming blocks of data, from the storage, through the accelerators, and back. Finally, the result of the query is returned to the software or stored back into the SSD.

The architecture of the proposed query processing engine is composed of five major components: **(1)** software extensions for DBMS in the host, including the Data Address Table (DAT) and the CStore Foreign Data Wrapper (FDW) extension of PostgreSQL, to manage the transfer of instructions and data, respectively, **(2)** data storage units, i.e., SSD, DDR-3, and host that are used as the primary or secondary database storage units and device controller cores, i.e., PCIe-3, DDR-3, SATA-3 to manage the data transfer to/from storage units, **(3)** a set of efficient database accelerators, i.e., filter, arithmetic, aggregation, group by, hash probe, hash build, and sort that are organized in a unidirectional ring bus, which is called RingBus of Accelerators (RBAA), **(4)** Programmable Interconnection Unit (PIU) to set up a path to transmit the data in a fully flexible fashion. It is composed of ***i)*** a 4-port bidirectional programmable data connection switch (PDCS) to exchange the data among SSD, DDR-3, host, and RBAA, ***ii)*** an arbiter to manage the DDR-3 concurrent requests, and ***iii)*** a set of synchronizing First In First Out (FIFO) modules for each individual port, separately for read and write directions, to cross the different clock domains, **(5)** Data and Process Controller (DPC) that is composed of an Instruction Cache (IC) to locate the instruction set and an execute Finite State Machine (FSM) ***i)*** to manage the accesses to the off-chip data sources and ***ii)*** to control the accelerators to execute the corresponding query, by issuing the appropriate control signals to the PIU and to the RBAA, respectively. These signals are generated by translating our query-specific instructions.

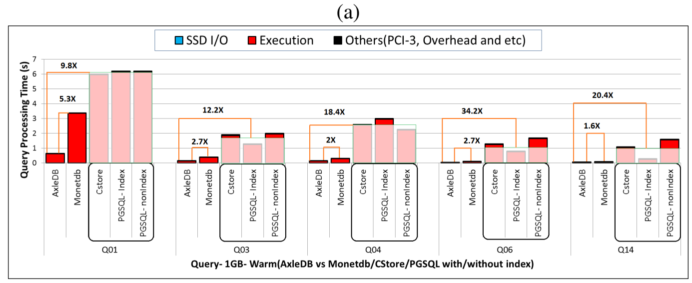
III. Experimental Results

We developed our engine on a VC709 FPGA development board with an XC7VX690T FPGA and 4GB of DDR-3 RAM. It accesses a Crucial M4-256GB SSD through a customized version of an SATA-3 controller, based on Groundhog [XX]. We evaluated our engine against the query processing engines of several state-of-the-art software DBMS: ***(i)*** MonetDB 11.21 as a popular column-oriented database system, ***(ii)*** PostgreSQL 9.5 (PGSQL) as a popular object-relational row-oriented database system, and ***(iii)*** CStore as the PostgreSQL's column-oriented data store extension. We evaluated our engine with five decision-support TPC-H queries, under various conditions. The studied queries are Q01, Q03, Q04, Q06, and Q14, which heavily utilize and stress the various hardware accelerators.

- **process-intensive queries** (Q01), as it can be seen in Figure 2(a), the I/O time of SSD is negligible and the execution time is dominating. In process-intensive workloads, the performance gain of the proposed FPGA-based engine is mainly the consequence of exploiting highly efficient query accelerators, in a deeply pipelined fashion.
- **I/O-process-balanced queries** (Q03, Q04), the improvement of the proposed engine is the consequence of both I/O efficiency and faster execution. For instance, the FPGA-based engine reduces SSD I/O time by 17.9X and execution time by 31.5X for Q04, in comparison to the index-enabled PostgreSQL, which leads to a total of 23.4X speedup for this particular case.
- **I/O-intensive benchmarks** (Q06, Q14), SSD I/O time is dominating. Comparing PostgreSQL with index-enabled vs. non-index versions, we unexpectedly observed a significant overhead of B-Tree indexes, which causes substantial performance degradation. In contrast, our FPGA-Based engine, CStore, and MonetDB, thanks to their column-oriented data storage, significantly reduce SSD I/O transfers. The results demonstrate that the FPGA-Based engine can process these queries, on average 2.4X faster than MonetDB, and 21.6X faster than the average of different versions of PostgreSQL.



(a)



(b)

Fig. 2. Total query processing time of the studied benchmarks in cold (a) and warm (b) modes, comparing the proposed FPGA-based engine (AxleDB) vs. MonetDB, CStore, and PostgreSQL in 1GB scale. Lower is better.

In summery, the significant speedup of the proposed FPGA-based engine against software-based comparison cases is the consequence of two optimization points: ***i)*** offloading the query processing onto the FPGA and following the streamline data flow execution model and ***ii)*** optimized accesses to the SSD (tightly coupled to the processing units -accelerators- in the FPGA). As mentioned earlier, these points have proportionally affected for each individual query,

References

[1] Jun, Sang-Woo and others. BlueDBM: an appliance for big data analytics. Proceedings of ISCA, pages=1–13, 2015, ACM.

[2] Casper, Jared and Olukotun, Kunle. Hardware acceleration of database operations. Proceedings of FPGA, pages=151–160, 2014, ACM.

[3] Woods, Louis and others. Ibex: an intelligent storage engine with support for advanced SQL offloading. Proceedings of the VLDB Endowment, volume=7, number=11, pages=963–974, 2014.