Lessons Learned Building Cross-Architecture Analytical Engines

Ivan Donchev Kabadzhov
Supervised by Raja Appuswamy
EURECOM
Biot, France
ivan.kabadzhov@eurecom.fr

ABSTRACT

Modern server hardware is increasingly heterogeneous, featuring a diverse mix of XPU architectures deployed across multi-vendor CPUs, GPUs, and FPGAs. Historically, database and systems developers have had to rely on either proprietary, architecture-specific solutions or low-level frameworks, which can complicate development and limit broader hardware adoption. This thesis explores hardware-software co-design for data-intensive applications, targeting the unification of programming models using open standards and exploring experimental techniques for automated query synthesis. We apply these approaches by progressing from dynamic relational data warehouses to computational genomics pipelines.

First, to tackle the engineering complexity of database workloads, we present SYCLDB, an open-source, portable analytical engine. We then investigate the "synthesize-versus-engineer" debate by using a fully autonomous Large Language Model (LLM) framework (SHADB) to synthesize query-specific GPU execution code. By using AI to establish an empirical performance ceiling, we isolate and lift generalizable optimizations back into SYCLDB. Second, extending this cross-architecture thesis, we present X-BQSR, a holistic redesign of genomic base quality score recalibration pipelines. X-BQSR verifies that hardware acceleration must encompass the entire data path, showing that I/O redesign is essential to prevent data movement from bottlenecking GPU gains. Finally, we establish the architectural performance limits of these designs on emerging RISC-V processors. Together, these systems demonstrate novel cross-architecture algorithms and holistic pipeline optimizations that pave the way for the next generation of scalable, open-standard analytics engines.

VLDB Workshop Reference Format:
Ivan Donchev Kabadzhov. Lessons Learned Building Cross-Architecture Analytical Engines. VLDB 2026 Workshop: VLDB Ph.D. Workshop.

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://gitlab.eurecom.fr/dislab/sycldb, https://gitlab.eurecom.fr/dislab/bqsr, and https://gitlab.eurecom.fr/dislab/SHADB.

Proceedings of the VLDB Endowment.
ISSN 2150-8097.

1 INTRODUCTION

The growing popularity of AI and analytics has led to an exponential increase in the data managed by enterprise and scientific institutions alike. Because general-purpose CPUs can no longer scale to meet the diverse demands of heterogeneous analytics workloads [3], there is a surging interest in hardware-accelerated engines. However, most contemporary solutions rely on closed, proprietary ecosystems that hinder portability across multi-vendor accelerators. To democratize data analytics, there is a pressing need for an open, standards-based, cross-vendor approach to hardware acceleration. SYCL has emerged as a leading solution in this space [12], enabling developers to write data-parallel applications once and execute them efficiently across diverse architectures.

To systematically evaluate the capability of open standards to power next-generation analytics, we present **SYCLDB** [7], an end-to-end analytical database engine. While prior GPU acceleration efforts rely on hard-coded, hardware-specific kernels (e.g., Crystal [14]) or proprietary software stacks (e.g., HeavyDB [4]), **SYCLDB** bridges this gap by mapping core relational operators to modular SYCL kernels integrated with a SQL frontend. The core insight of this work is demonstrating that system-level optimizations—rather than merely relying on SYCL's built-in compiler abstractions—can enable a single codebase to achieve performance competitive with highly specialized implementations across multi-vendor GPUs.

However, modern database workloads must accommodate an infinite space of potential query plans, which traditionally demands hand-crafted kernels tailored to specific hardware features. Inspired by CPU-centric synthesis systems like GenDB [8] and Bespoke OLAP [15], we study whether GPU analytics should rely on automatically generated, query-specific code or reusable engineered operators using an LLM-driven framework called **SHADB** [6]. Rather than relying on fragile end-to-end prompting, **SHADB** employs a structured code validation mechanism to verify the correctness of LLM-generated physical query plans. By analyzing the performance gap between AI-synthesized code and engineered engines, we extract generalizable compiler insights that can be lifted back into **SYCLDB**, providing a concrete roadmap for automated, portable query optimization.

We further extend these principles beyond relational engines into the domain of precision medicine. In computational genomics, Base Quality Score Recalibration (BQSR), which corrects systematic sequencing errors, consumes over half of the data preprocessing time [10]. Crucially, this pipeline closely mirrors relational operations like selections, projections, and grouped aggregations over massive array structures. Hence, we present **X-BQSR**, a holistic, cross-architecture SYCL implementation of the BQSR pipeline. **X-BQSR** natively extends our core system design optimizations to the

storage layer, demonstrating that resolving I/O bottlenecks alongside parallel compute can outperform proprietary, closed-source genomics pipelines like NVIDIA Parabricks [11].

In summary, this thesis advances portable parallelism for data-intensive applications through three interconnected efforts: providing a performance-portable analytical foundation via **SYCLDB**, exploring automated, verifiable query compilation using **SHADB**, and demonstrating genomic pipeline acceleration via **X-BQSR**. By evaluating these systems on emerging open instruction sets like RISC-V alongside standard GPUs, this work provides a comprehensive blueprint for the future of cross-architecture data systems.

2 PORTABLE RELATIONAL ANALYTICS WITH SYCLDB

To systematically evaluate the capability of open standards to power next-generation analytics, we present **SYCLDB** [7], an end-to-end analytical database engine built upon the cross-architecture SYCL C++ programming model. Traditional state-of-the-art acceleration solutions represent a trade-off: Crystal [14] lacks a full relational SQL frontend and relies entirely on pre-engineered, hand-written tiled GPU primitives, whereas HeavyDB [4] is as a production-grade, full-fledged engine utilizing an LLVM-JIT infrastructure to compile queries into execution kernels on the fly. To bridge this gap, **SYCLDB** integrates a full SQL query engine via a decoupled client-server architecture: an Apache Calcite-based frontend parses and optimizes relational plans, which are transmitted via Apache Thrift RPC to a high-performance C++ execution engine that leverages Apache Arrow for efficient in-memory columnar ingestion.

At the execution layer, **SYCLDB** maps relational operators (Selection, Projection, Join, and Aggregation) to data-parallel, kernel-at-a-time SYCL abstractions. To rigorously test the absolute cross-architecture performance portability of this baseline engine, we evaluated its execution behavior on emerging RISC-V instruction set architectures supporting RISC-V Vector (RVV) extensions alongside standard hardware targets. Our profiling revealed that by compiling the same unified codebase using architectures like AdaptiveCpp, the open-standard implementation introduced zero noticeable abstraction overhead relative to native vendor-specific toolchains. Driven by the native `sycl::reduction` trees and pointer-free linear-probing hash arrays, the execution trace achieved structural hardware saturation. Roofline models confirmed that the baseline operators successfully hit the physical memory-bandwidth boundaries of the underlying target hardware, establishing a highly performant, portable relational core across multi-vendor GPUs, and RVV accelerators.

3 BRIDGING SYNTHESIZED AND ENGINEERED GPU EXECUTION

Natively executing dynamic queries on accelerators remains difficult due to complex query-plan combinations and manual memory tracking. We experimentally investigate LLM code synthesis for GPU databases using a framework called **SHADB** [6]. Rather than prompting with raw SQL for one-shot code generation, **SHADB** provides the declarative query together with the target dataset's physical layout. The model must return a dual payload: a structured physical query plan that fixes data mappings and join/aggregation bounds, followed by matching executable C++ containing the CUDA or HIP kernels. Committing to the plan before generating code acts as a structured chain-of-thought step that stabilizes the design and reduces logical errors. **SHADB** then runs a deterministic Profile-Guided Optimization (PGO) loop with three gates: binary validation, device-only execution audits (rejecting host-CPU fallbacks), and comparison with a relational oracle. The loop requires no human-in-the-loop engineering: **SHADB** converts profiler counters—including effective DRAM bandwidth, cache hits, register pressure, and warp divergence—into readable optimization directives and appends them to the model history before the next edit, compile, and profile iteration.

Using this framework on the Star Schema Benchmark (SF100), we established an empirical performance ceiling. The synthesized code successfully hit the hardware memory-bandwidth limit, outperforming state-of-the-art JIT-compiled GPU engines like HeavyDB by 7.4× and expert hand-coded systems like Crystal by 2.6×. Reaching the reported performance ceiling through automated synthesis and successive profile-guided optimization loops required a total budget of 4.2 GPU-hours and $126 for the entire benchmark suite using Anthropic's Claude Opus 4.8. In addition, portability was poor: code synthesized at SF10, SF100, and SF200 could not directly be transferred between scale factors, requiring scale-specific resynthesis and tuning.

To determine if a general-purpose engine could bridge this gap, we systematically decomposed the AI-generated performance gains to isolate generalizable optimizations from workload-specific hacks (e.g., L2-resident bitmasks). We then lifted these generalizable optimizations back into our portable **SYCLDB** engine:

Composition-Based Kernel Fusion: We utilized modern SYCL compiler capabilities (such as AdaptiveCpp's [1] dynamic functions) to stitch pre-compiled, type-safe SYCL operator blocks together at launch time. Unlike HeavyDB's heavyweight runtime JIT fusion—which must parse query graphs, manipulate complex Intermediate Representations (IR), and invoke full compiler toolchains to emit device assembly on the fly—this eliminates intermediate global memory materialization overheads entirely while bypassing the long latency and complexity of runtime compilation.

Hardware-Specific Shared Memory Tiling: Dynamically binding heavily reused hash tables and intermediate state variables to low-latency local/shared memory clusters instead of device VRAM to eliminate redundant global memory round-trips.

Coalesced Memory Alignments and Asymmetric Batching: Reorganizing parallel thread-blocks and register-level inter-operator dataflow to strictly match the underlying physical hardware warp constraints, guaranteeing fully coalesced reads and optimizing register occupancy.

Our evaluation in Figure 1 demonstrates that the fully optimized **SYCLDB-Opt** engine successfully reaches within 1.27× of the specialized AI ceiling. Furthermore, the identical optimized **SYCLDB** codebase executes seamlessly across both NVIDIA L40S and AMD Instinct MI210 GPUs. This work concludes that engineering wins on GPUs: optimized general-purpose engines can rival synthesized AI ceilings while retaining hardware portability and avoiding the

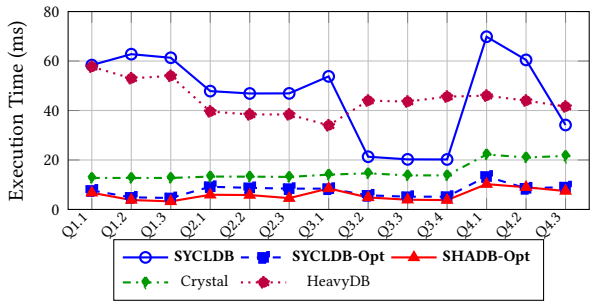

Figure 1: SSB SF100 query execution time on NVIDIA L40S (kernel time, warm run).

financial costs of single-use query synthesis. LLMs, therefore, are best utilized as a discovery tool for structural optimizations that portable engines can readily adopt.

4 HOLISTIC ACCELERATION OF COMPUTATIONAL GENOMICS WITH X-BQSR

We apply the co-design principles developed for **SYCLDB** to accelerate the Genome Analysis Toolkit (GATK) pipeline [10]. Base Quality Score Recalibration (BQSR), the dominant preprocessing bottleneck before variant calling, uses empirical error models to correct systematic machine-sequencing biases. From a database perspective, BQSR executes in two relational phases over terabyte-scale aligned reads (BAM files):

(1) `BaseRecalibrator`: A heavy multi-dimensional `GROUP BY` aggregation that computes empirical error frequencies across covariate keys (read group, reported quality, cycle, and dinucleotide context).
(2) `ApplyBQSR`: A streaming map/update operator that evaluates a hierarchical Bayesian model via dynamic lookups to rewrite read quality scores.

An end-to-end performance analysis reveals that GPU acceleration cannot focus solely on device compute kernels; format handling, including BGZF block decompression and string parsing, routinely dominates execution time. Prior FPGA-based approaches [9] accelerated only `BaseRecalibrator` arithmetic while neglecting `ApplyBQSR` and the surrounding data path, causing severe pipeline stalls. **X-BQSR** resolves this by redesigning both parallel compute operators and asynchronous I/O ingestion into a single, portable SYCL framework.

At the compute layer, **X-BQSR** maps BQSR's sequential string operations to SIMT hardware. Sequence alignment clipping is converted into coalesced per-read updates via a Structure-of-Arrays (SoA) layout. Feature detection uses a branch-aware scan over reference coordinates, avoiding intermediate vector materialization to minimize global memory bandwidth. For covariate aggregation, thread-block local shared-memory histograms eliminate global memory write contention before merging state into the global summary table. Finally, `ApplyBQSR` caches recurrent Bayesian estimates in a 1 MB device-side lookup table per read group to convert arithmetic bottlenecks into high-throughput memory queries.

At the data-path layer, double-buffered CPU host readers stream standard BAM/BGZF chunks directly to SYCL compute workers across NVIDIA, AMD, and Intel architectures. Asynchronous serialization and compression loops hide device-to-host completion latencies.

As shown in Figure 2, against the single-threaded GATK gold standard, **X-BQSR** achieved up to a 141.7× speedup on production-scale datasets. Furthermore, it consistently outperformed NVIDIA's proprietary closed-source Parabricks [11] by 3.6× to 8.7×.

5 EXPLORING THE FUTURE: RISC-V ACCELERATION

While evaluating multi-vendor GPUs proves the commercial readiness of SYCL, we also look to the future of fully-open hardware architectures. The RISC-V Instruction Set Architecture (ISA), particularly the RISC-V Vector Extension (RVV), has emerged as a viable new standard for data-intensive workloads. Crucially, SYCL provides a direct, high-level programming model to communicate with and offload computations to these novel architectures.

To understand the current state of RISC-V processors, we utilized a Cache-Aware Roofline Model (CARM) [5] to analyze query limits on the commercially available 64-core SG2042 RISC-V CPU. Our CARM analysis confirmed that our manually unrolled scalar implementations successfully hit the theoretical peak GFLOPS and memory bandwidth limits of the CPU's architecture. Establishing this verified scalar baseline clarifies where the observed benefits of vectorization originate: surpassing these physical architectural walls requires exploiting the RISC-V Vector Extension (RVV).

To evaluate vectorization and execute **SYCLDB** queries on modern, ratified RVV 1.0 extensions, we leveraged the SYCLARA platform [13], an open hardware-software platform integrating the CVA6 RISC-V core [16] with ARA2 [2] (an open-source RVV 1.0 implementation). We seamlessly offloaded SYCL kernels to this FPGA-based platform utilizing the oneAPI Construction Kit (OCK). Using this infrastructure, we achieved RVV 1.0 acceleration on **SYCLDB** operators, with the relational projection operator demonstrating up to a 2.8× speedup over CVA6 scalar execution. Finally, we demonstrated the extensibility of this fully-open co-design stack by evaluating **X-BQSR** performance on the RISC-V accelerator. This underscores that a unified, standards-based programming model can bridge the gap between commodity multi-vendor GPUs and emerging open-hardware platforms for both relational databases and genomic pipelines. However, fully realizing this open hardware-software paradigm requires shifting from isolated node acceleration to system-wide coordination. This operational shift forms the foundation of our long-term deployment strategy, which we detail next as a structured research roadmap.

6 FUTURE ROADMAP AND CONCLUSION

By unifying the architectural portability of open programming frameworks such as SYCL and RISC-V with autonomous LLM profiling, this thesis establishes a blueprint for next-generation data-intensive systems. **SYCLDB** and **X-BQSR** already rival or exceed proprietary, vendor-locked stacks; the dissertation's final stage advances three interconnected research vectors.

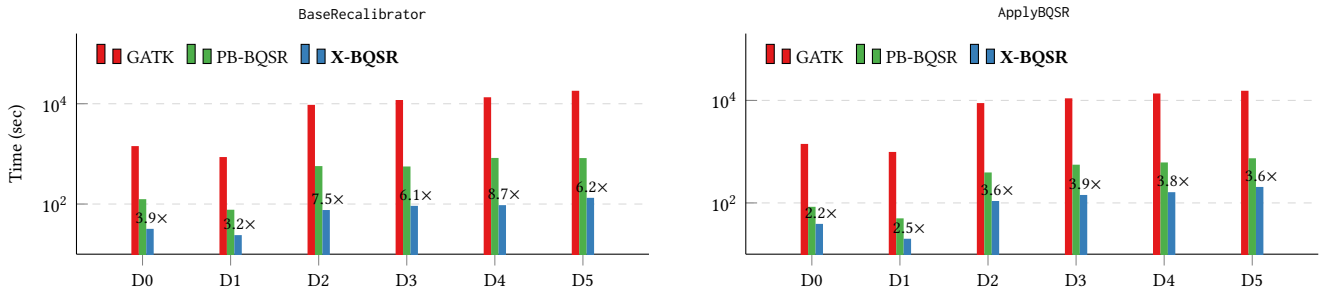

Figure 2: Execution time comparison of X-BQSR against GATK and NVIDIA Parabricks (D0–D5). Speedup values (×) are relative to NVIDIA Parabricks.

Generalizable Computational Storage and Data Lakes. Building on **X-BQSR**'s holistic I/O and serialization redesign, we will extend **SYCLDB** from an in-memory relational engine into a distributed data lakehouse engine. Abstracting its asynchronous, double-buffered parsing pipeline to query open table formats will enable evaluation of portable computational-storage offloads that reduce data movement in out-of-core environments.

Autonomous Heterogeneous Co-Orchestration. **SHADB**'s PGO loop will evolve from a single-GPU kernel compiler into a synthesizer of cluster-wide execution topologies. The goal is to navigate CPU–GPU and multi-GPU orchestration spaces, reasoning about register-occupancy cliffs, warp divergence, and asymmetric PCIe latencies to discover scheduling policies across divergent hardware.

End-to-End Open Genomics Frameworks. We will expand **X-BQSR** from one accelerated preprocessing step into a fully open-source, vendor-neutral GATK pipeline. Porting and fusing sequence alignment, variant discovery, and the remaining genomics stages into one open-standard runtime will test portable execution across commodity multi-vendor accelerators and emerging RISC-V systems.

Ultimately, this work demonstrates that the trade-off between performance and portability is a false dichotomy. Engineered, open-standard engines remain the most effective vehicle for sustainable hardware acceleration, while automated synthesis serves as a powerful discovery engine to uncover hardware-specific performance ceilings. By systematically lifting AI-discovered optimizations into reusable, cross-architecture frameworks, we eliminate vendor lock-in without sacrificing bare-metal efficiency—paving the way for fully open, scalable analytics across heterogeneous hardware.

ACKNOWLEDGMENTS

This thesis was supported by the European Union's Horizon 2020 programme (Grant No. 101136962), the Horizon Europe SYCLOPS project (Grant No. 10109287), UK Research and Innovation (UKRI) under the UK Government's Horizon Europe guarantee (Grant Nos. 10098097 and 10104323), and the Swiss State Secretariat for Education, Research and Innovation (SERI).

REFERENCES

[1] Aksel Alpay and Vincent Heuveline. 2023. One Pass to Bind Them: The First Single-Pass SYCL Compiler with Unified Code Representation Across Backends. In *Proceedings of the 2023 International Workshop on OpenCL* (Cambridge, United Kingdom) *(IWOCL '23)*. Association for Computing Machinery, New York, NY, USA, Article 7, 12 pages. https://doi.org/10.1145/3585341.3585351

[2] M. Cavalcante, F. Schuiki, F. Zaruba, M. Schaffner, and L. Benini. 2020. Ara: A 1-GHz+ Scalable and Energy-Efficient RISC-V Vector Processor. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 2 (2020), 530–543.

[3] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. 2011. Dark Silicon and the End of Multicore Scaling. *SIGARCH Computer Architecture News* 39, 3 (June 2011), 365–376.

[4] HEAVY.AI. 2025. HeavyDB. Archived at https://web.archive.org/web/20250108153216/https://www.heavy.ai/. https://www.heavy.ai/

[5] A. Ilic, F. Pratas, and L. Sousa. 2014. Cache-Aware Roofline Model: Upgrading the Loft. *IEEE Computer Architecture Letters* 13, 1 (Jan. 2014), 21–24.

[6] Ivan Donchev Kabadzhov, Eugenio Marinelli, and Raja Appuswamy. 2026. From Custom-Fit to Portable: Bridging the Gap Between Synthesized and Engineered GPU Query Execution. , 13 pages. https://doi.org/10.48550/arXiv.2607.07632 arXiv:2607.07632 [cs.DB]

[7] Ivan Donchev Kabadzhov, José Morgado, Aleksandar Ilic, and Raja Appuswamy. 2025. Open, Cross-Architecture Acceleration of Data Analytics with SYCL and RISC-V. In *Proceedings of the 23rd International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar)*. Springer, Dresden, Germany, 1–12. https://hal.science/hal-05507632v1

[8] J. Lao and I. Trummer. 2026. GenDB: The Next Generation of Query Processing — Synthesized, Not Engineered. *Proceedings of the VLDB Endowment* 19, 1 (2026), 1–14.

[9] M. Lo, Z. Fang, J. Wang, P. Zhou, M. C. F. Chang, and J. Cong. 2020. Algorithm-Hardware Co-design for BQSR Acceleration in Genome Analysis ToolKit. In *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, Fayetteville, AR, USA, 157–166. https://doi.org/10.1109/FCCM48280.2020.00029

[10] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M.A. DePristo. 2010. The Genome Analysis Toolkit: A MapReduce Framework for Analyzing Next-Generation DNA Sequencing Data. *Genome Research* 20, 9 (2010), 1297–1303.

[11] NVIDIA Corporation. 2020. NVIDIA Parabricks.

[12] Ruyman Reyes, Gordon Brown, Rod Burns, and Michael Wong. 2020. SYCL 2020: More than meets the eye. In *Proceedings of the International Workshop on OpenCL* (Munich, Germany) *(IWOCL '20)*. Association for Computing Machinery, New York, NY, USA, 1–2. https://doi.org/10.1145/3388333.3388870

[13] Mojtaba Rostami Bilandi, Ivan Donchev Kabadzhov, and Raja Appuswamy. 2025. SYCLARA: An Open Hardware–Software Platform for Evaluating SYCL Applications on RISC-V Vector Accelerators. In *Proceedings of the 13th International Workshop on OpenCL and SYCL (IWOCL '25)*. Association for Computing Machinery, New York, NY, USA, 1–2. https://doi.org/10.1145/3731125.3731136

[14] A. Shanbhag, S. Madden, and X. Yu. 2020. A Study of the Fundamental Performance Characteristics of GPUs and CPUs for Database Analytics. In *Proceedings of the 2020 International Conference on Management of Data (SIGMOD)*. ACM, New York, NY, USA, 1617–1632. https://doi.org/10.1145/3318464.3380595

[15] Johannes Wehrstein, Timo Eckmann, Matthias Jasny, and Carsten Binnig. 2026. Bespoke OLAP: Synthesizing Workload-Specific One-size-fits-one Database Engines. arXiv:2603.02001 [cs.DB] https://arxiv.org/abs/2603.02001

[16] F. Zaruba and L. Benini. 2019. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27, 11 (Nov. 2019), 2629–2640.