DPUF: DPU-accelerated Near-storage Secure Filtering

Narangerelt Batsoyol
UC San Diego
San Diego, USA

Daniel Waddington
IBM Research
Almaden, USA

Swaminathan Sundararaman
IBM Research
Almaden, USA

Steven Swanson
UC San Diego
San Diego, USA

As data volumes grow exponentially, the need for "data lakehouse" [2, 4] solutions that enable complex queries on exabytes of data is becoming acute. These systems store vast amounts of immutable data in columnar formats like Parquet [3] and leverage cloud object stores such as Amazon S3 [1] for their flexibility, scalability, and cost-efficiency. However, processing this data often leads to network bottlenecks, particularly when large datasets must be transferred over high-latency networks (e.g., WANs) for processing, as storage is decoupled from compute resources and data is geographically distributed. One way to alleviate this bottleneck is to reduce the amount of data transferred and only ship the portions the application needs. Yet, with client-side encryption, the entire dataset must be transferred to the client before filtering can occur, as illustrated in Figure 1 (left), leading to performance inefficiencies [5, 10].

Pushing data filtering closer to storage can alleviate these bottlenecks. Services like Amazon S3 Select [9] partially push computations down, but cannot filter encrypted data without the client's key.

We introduce *DPUF*, a near-storage data filter using a Data Processing Unit (DPU) as a secure enclave. A DPU is a programmable SoC with power-efficient cores, high-bandwidth networking, and built-in accelerators [8], connected via PCIe as a "Super SmartNIC." It offers secure networking and computational capabilities.

DPUs support secure boot and remote attestation to ensure that only verified software runs on the DPU. Physical isolation and robust security features create a minimal attack surface for processing sensitive data in the cloud.

By harnessing the DPU's hardware accelerators for de-/re-encryption and its compute cores for filtering, DPUF reduces data movement, lowers latency, and preserves client-side encryption, all while minimizing client resource usage as shown in Figure 1 (right).

The key contributions of this paper are:

- **DPUs for Data Filtering:** We apply DPUs to near-storage data filtering tasks and demonstrate how they can offload and accelerate filtering operations.
- **An Optimized Data Filtering Pipeline:** We introduce an optimized pipeline architecture within the DPU that efficiently handles multiple stages of data processing, from decryption to final output.

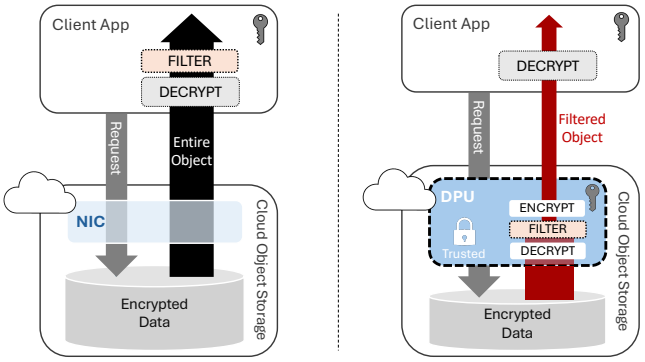

Figure 1. In traditional data lake storage (left), entire encrypted objects are transferred to the client for filtering. With DPUF(right), the DPU securely filters data close to storage, sending only filtered data to the client.

- **Efficient Task Management for Cryptographic Operations:** We detail how DPUF efficiently uses the DPU's cryptographic engine, addressing task scheduling and memory management challenges to maximize the decryption throughput.
- **Seamless Integration with Existing Storage Infrastructure:** DPUF performs push down without changing the S3 Select REST API.

The following sections briefly describe DPUF's design and present some evaluation results.

DPUF's Design

DPUF accelerates client queries by securely filtering data from large, encrypted datasets stored in cloud-based object storage. By offloading data filtering tasks to the DPU, DPUF minimizes the time and network resources required to extract valuable insights, especially for clients operating over bandwidth-limited networks. Like Amazon S3 Select, DPUF is designed for single-object queries, processing one file (or object; we use the terms interchangeably) per request.

Figure 2 depicts the usage scenario that DPUF targets: the client connects over a slow network (e.g., a WAN) to a cloud server equipped with a DPU. The cloud server hosts large encrypted objects that the client needs to access, but downloading the entire dataset for *client-side filtering* would be inefficient. Instead, the client submits filtering requests to DPUF, which processes the data on the DPU and returns

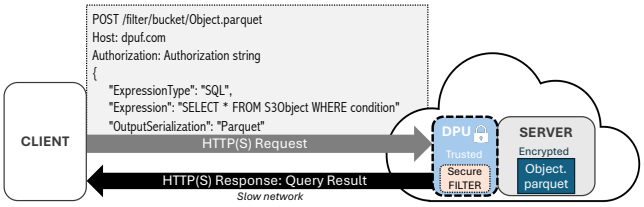

Figure 2. Overview of DPUF . The client submits an HTTP(S) POST request to the DPU, which retrieves encrypted data from cloud-based object storage, filters the data based on the client's query, and returns the results.

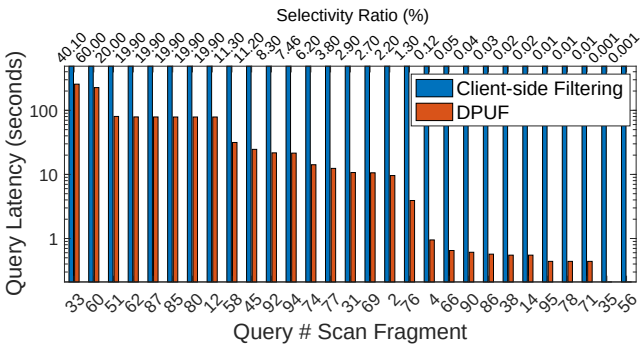

Figure 3. Query latency across different fragments of the TPC-DS `web_sales` table ordered by decreasing selectivity ratio. Latency are presented on a logarithmic scale to highlight the significant differences in performance.

the results, reducing the overhead of data transfer and the computational burden on the client.

To address the challenges, DPUF implements a pipeline that breaks the data filtering process into modules, such as data retrieval, decryption, filtering and repackaging. This design optimizes resource usage and ensures efficient data flow throughout the pipeline.

DPUF uses a DPU's ARM cores and cryptographic accelerator to decrypt and filter data, sending only the filtered results back to clients. This approach preserves confidentiality, leveraging the DPU as a trusted enclave and managing cryptographic keys internally.

DPUF works with any S3-compatible storage by exposing the DPU's IP as an endpoint for SQL-based filter requests via `HTTP(S) POST`. Clients specify filtering conditions using `SELECT` statements with `FROM`, `WHERE`, and `LIMIT` clauses, retrieving just the needed data instead of entire objects.

Upon receiving a request, DPUF extracts the object name, query, and credentials, then retrieves the encrypted object from storage. The DPU decrypts the data using its crypto engine, applies the query, and sends the filtered results to the client.

However, DPUs have limited compute and memory resources. The cryptographic engine accelerates decryption but requires buffer mappings that introduce overhead when repeated frequently. To address these issues, DPUF implements a pipeline for data retrieval, decryption, filtering, and repackaging—optimizing resource usage and ensuring efficient data flow.

Evaluation

In this section, we evaluate the performance of DPUF against traditional client-side filtering for encrypted data and S3 Select for unencrypted data. Our primary focus is to understand how DPUF accelerates query latency, reduces data transfer, and minimizes network cost.

Figure 2 illustrates our system for testing DPUF, featuring a server with a DPU and a client connected over a slow network. The server is hosted on an x86 machine running MinIO [7], an open-source, high-performance object storage system compatible with S3 APIs, which serves Parquet files as objects. The server and client are equipped with dual 24-core Intel Cascade Lake processors and 192 GB of DRAM.

DPUF's main benefit is in reducing bandwidth consumption and query latency. The benefits it provides grow the selectivity of the query. Figure 3 shows the query latency using all scan fragments from the `web_sales` table of 742 MB, ordered by selectivity ratio ranging from 40% to 0.001%. The evaluations were performed over a 400 Mbps link between the client and the server. DPUF shows 452× speedup in average ranging from 2× to 2322× across the queries.

For most queries, the relevant data represents only a small subset of the entire object, and DPUF is most effective when the relevant subset is under 20% of the original data. This observation aligns with studies of analytics workloads, which show that applications typically use only 20% of the data they retrieve, leading to substantial inefficiencies in data transfer [6].

We have also evaluated DPUF's performance sensitivity to link bandwidth, explored the details of where time goes during processing, and examined the cost savings due to reduced network traffic.

References

[1] Amazon s3. https://aws.amazon.com/s3/. Accessed: 2024-07-30.
[2] Amazon Web Services. What is a data lake?, 2024. Accessed: 2024-06-30.
[3] Apache. Parquet, 2023. Accessed: 2024-06-30.
[4] Armbrust, M., Ghodsi, A., Xin, R., and Zaharia, M. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In *Proceedings of CIDR* (2021), vol. 8, p. 28.
[5] Durner, D., Leis, V., and Neumann, T. Exploiting cloud object storage for high-performance analytics. *Proceedings of the VLDB Endowment 16*, 11 (2023), 2769–2782.
[6] Microsoft. Data lake storage: Query acceleration. https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-query-acceleration, n.d. Accessed: 2024-07-28.
[7] MinIO. Minio high performance object storage, 2023. Accessed: 2024-06-30.
[8] NVIDIA. What's a dpu? the data processing unit, 2023. Accessed: 2024-06-30.
[9] Services, A. W. Selecting content from objects, 2023. Accessed: 2024-06-30.
[10] Seth, B., Dalal, S., Jaglan, V., Le, D.-N., Mohan, S., and Srivastava, G. Integrating encryption techniques for secure data storage in the cloud. *Transactions on Emerging Telecommunications Technologies 33*, 4 (2022), e4108.