High Performance Open Source Database Architecture

rENIAC Data Engine for Cassandra NoSQL on Dell EMC Infrastructure

Abstract

This document evaluates the performance advantages of running an Apache® Cassandra® NoSQL database using rENIAC's hardware acceleration framework on Dell EMC PowerEdge R740 servers with Intel® Xeon® Scalable processors and Arria® 10 GX FPGAs. This reference architecture demonstrates how rENIAC Data Engine (rDE) improves throughput and reduces latency for existing databases.

August 2020

Table of Contents

Executive summary

WHY

Data is growing exponentially and continues to put a massive burden on existing infrastructure. In a time when Moore's Law is breaking down, the demand for higher data processing performance in the data center continues to grow. The enterprise can no longer count on the typical performance CAGR that has been traditionally available with each new release of CPUs. To meet the performance demands, the industry has started to adopt a more heterogeneous approach to computing by leveraging accelerators such as FPGAs. However, with more sophisticated architectures comes added complexity. This is the reason why Dell Technologies, rENIAC® and Intel have teamed up to offer a solution that adds significant performance to Apache Cassandra databases while removing the complexity of standing up a complete solution from scratch.

WHAT

In this solution, the Dell EMC PowerEdge R740 servers provide a perfect balance of accelerator cards, storage and compute resources optimized for workload acceleration. rENIAC's Data Engine accelerates Apache® Cassandra® databases by leveraging Intel Programmable Acceleration Cards and the Acceleration Stack for Intel Xeon® Scalable processors with Intel Programmable Acceleration Card (PAC) with Arria® 10 GX FPGAs. The solution from rENIAC built on Dell EMC servers with Intel accelerators offers seamless integration with improved performance, while enabling portability across multiple Intel FPGA platforms and Dell EMC server configurations.

HOW

Dell EMC, Intel and rENIAC bring together an acceleration solution that solves many of the bandwidth and latency problems that plague Cassandra NoSQL databases. Intel FPGA PAC cards are available with in a variety of Dell EMC PowerEdge servers and have been thoroughly tested and validated, providing a solid foundation to run the entire rENIAC stack. This allows organizations to easily integrate the complete solution with the peace of mind that they are running on highly optimized and fully validated configurations, enabling them to focus on the business problems that matter most to them while drastically reducing risk.

Application data growth has outpaced Moore's Law

DATA VOLUME INCREASE IN CUSTOMER-FACING APPLICATIONS

In this data-centric age, massive and ever-increasing amounts of data are gathered, decomposed, merged, analyzed, correlated, trended, personalized, anonymized and presented.

Data growth is especially fast and unbounded when related to customer-facing applications. This is due to a combination of an increasing number of customers over time multiplied by increasing data per customer.

At the same time, business success is often a function of application decision response times. This is occurring at a rate that Moore's Law cannot keep up with. Hardware is becoming increasingly complicated and costly as engineers and architects continue to try to keep up with increasing real-time demands.

As business success is often a function of application decision response times, innovative organizations are driving success by efficiently dealing with ever-increasing data volumes and application usage.

Traditional methods to support data growth

Database architectures and techniques have been evolving in an attempt to meet the needs of today's applications that involve large numbers of users, huge and ever-increasing amounts of data, and stringent latency requirements. The details vary by database type and data model, but most architectures involve creating clusters of database nodes to spread the user demand for data evenly and avoid a specific node becoming overloaded and, consequently, delivering poor performance. NoSQL, SQL, key value, in memory, cache and graph databases are all available as open source solutions based on their strengths and use cases in which they perform best.

That sounds good, but in practice, a lot of money and energy can be spent deploying these architectures only to achieve poor and ever-degrading performance at scale.

THE CASSANDRA EXAMPLE

Take the NoSQL database Cassandra as an example. Cassandra was architected for performance-at-scale via a powerful distributed architecture. Cassandra implements a method for striping data across nodes with the ability for clients to "learn" where data exists to direct read requests to the appropriate server nodes serving the desired data. In theory, this allows many compute nodes to perform efficiently as a single virtual cluster to service user requests.

However, as Cassandra has been deployed for several years in performance-sensitive, data-centric applications, a few consistent observations can be made:

- Data always increases. As the amount of data increases, the clusters keep getting larger and larger, with no end in sight. This results in more hardware costs to deploy more nodes, more administrative costs to manage the nodes and continuous manual tuning of data partitioning algorithms. This means that clusters can become 1,000s of nodes large in the attempt to meet performance demands.
- Even if you build nodes on top of the fastest (and most expensive) server, it remains difficult or impossible to meet predictable/deterministic millisecond (ms) or sub ms response time service level agreements (SLA) obligations at peak load.

- The process for partitioning data across many nodes with the goal of achieving better throughput and latency can be time consuming and inexact. If you accidently create “hot spots” of data so that data requests are not evenly spread across the nodes, you can harm performance no matter how little data supported per node. As data grows and as user patterns evolve, the data partitioning approach that worked well yesterday may not work nearly as well today.

As a result, managers of Cassandra clusters may have to make tradeoffs between latency, equipment costs, and the time required to continually try to squeeze out more performance from a moving target. Cassandra specialists or experts who have spent years learning the intricacies of this particular database are hired to keep up with the consistent tuning and optimization requirements, and even then, these problems persist. And that is for Cassandra, which is distributed, built to scale and always on; the situation can be even worse for other databases.

Why is it still hard to process so much data?

Modern databases are built to use commodity hardware efficiently, specifically input/output (I/O) and processing components. They are designed to rely on horizontal scaling to scale (query) throughput while ensuring commonality in all of the nodes in the cluster. Large scale data-centric environments often have applications with read/write ratios ranging anywhere from 5:1 to 500:1, exacerbating this inefficiency.

What we are observing here is not a weakness in the databases themselves. The problem is that databases perform functions — in fact the most common and performance sensitive functions — that do not execute well on a standard CPU-only server. Traditional CPU architectures maximize instructions executed per clock cycle (IPC). The techniques used to achieve high IPC — caching of code and data, deep pipelines, register renaming, and branch prediction; all contribute to good performance for the background tasks that databases execute. This means that the inability to process so much data in real time is a reflection of the limitations of the underlying traditional server architecture.

At rENIAC, we profiled different, commonly used modern databases — MongoDB®, RocksDB®, Cassandra®, and HBase® to understand I/O and processing behavior. We found three areas where databases struggle and end up consuming a disproportionate amount of CPU resources, namely:

- Storage I/O
- Network I/O
- Computational complexity

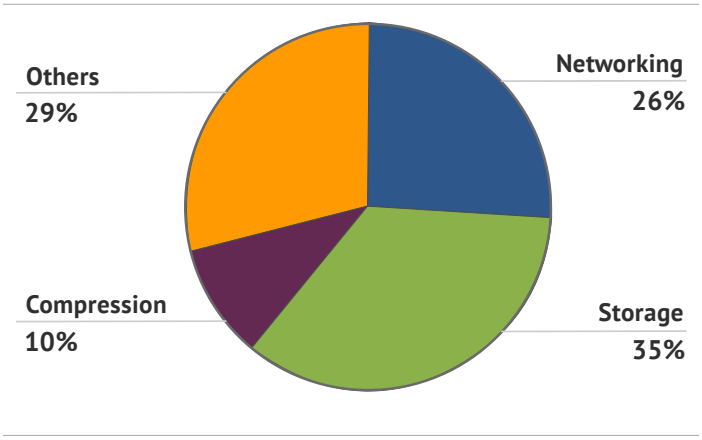


Figure 1. CPU usage: Read-heavy workload

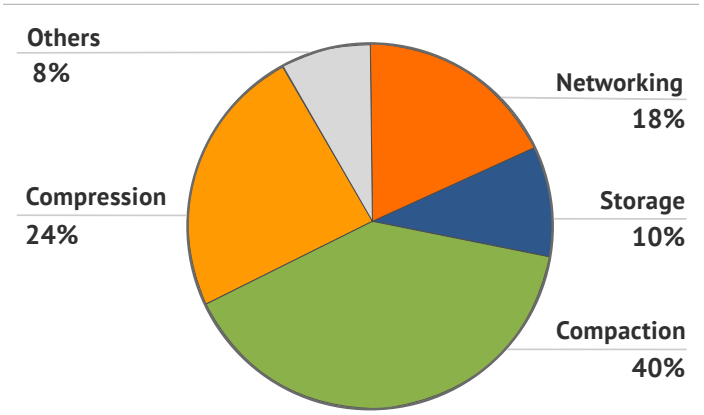


Figure 2. CPU usage: Write-heavy workload

Figure 1 and Figure 2 break down the CPU usage of Cassandra for both read-heavy and write-heavy workloads to illustrate that inefficiencies are easily recognized on both ends of the spectrum.

Figure 1 shows a summary of CPU cycles spent on open source databases for a typical read-heavy workload. In an architecture where the database is accessing flash memory and database logic is a crucial operation, significant CPU resources are required just to keep up with a fast flash storage drive — 35% of the CPU usage profile in this workload goes to storage I/O alone. Similarly, with network I/O, 25% of the CPU usage profile is consumed, driven by the overhead of networking (TCP/IP) for a standard 10Gbps network interface. Together, storage and network I/O consume 60% of CPU usage; add in compression and 70% of CPU resources are spent on operational activities.

Figure 2 shows a summary of where CPU cycles are spent on open source databases for a typical write-heavy workload. Modern databases rely on computationally complex functions such as compression, encryption, and compaction, which add to computational load. It has been well established that both compression and encryption are computationally expensive, and it is even prohibitive for CPUs to carry out these functions. For a write heavy workload, compaction and compression/decompression together account for almost 64% of the total CPU cycles.

In summary, our profile showed that combination of system compute tasks and I/O operations consume up to 92% of the total CPU cycles.

These research findings point to two main points:

1. Standard open source database software has reached an optimization threshold through the use of general-purpose CPUs and SSDs, where data growth has outpaced the ability to process it within SLA (service level agreement) windows to meet target latency requirements.
2. An advancement in data workload acceleration is required and can be achieved through software-hardware optimization.

These inherent inefficiencies mean a lot of CPU cores are required to handle some basic functions, particularly as data scales. Data-centric database workloads, no matter how efficiently coded, perform badly on traditional compute-optimized environments.

One of the real core functions that a database is intended to execute is read servicing, but, in databases like Cassandra, they are not optimized to do so despite being used in read-heavy use cases. Now, with the rise of read-intensive workloads used across industries like finance, ecommerce and security like user profile stores, recommendation engines, fraud prevention, and AI or ML modeling, enhanced read performance is critical. Many IT personnel refer to the issue as the “read problem.” Often, a cache or in-memory database solution is added in front of a persistent database like Cassandra in an effort to combat the read problem, but the same issues of speed and scale persist with this architecture. Many different database types may also be employed to handle different data workloads and models, but also often require a caching layer.

In the rest of this document, we use the “read problem” to demonstrate how an approach that combines hardware and software, like rENIAC Data Engine, can help organizations to achieve performance and latency requirements for use cases like those just mentioned using Cassandra as the database of example.

Tuning open source databases for maximum performance

If the performance of databases is being negatively affected by a traditional CPU infrastructure for the most common and the most user-perceptible task (read); why not relegate that task to something much more efficient at executing it?

rENIAC Data Engine tunes the read inefficiency inherent to open source databases to increase overall throughput and significantly reduce latency. One method of doing so is through the use of commercially available FPGAs, like those available from Intel, that are programmed to provide hardware assist for many of the functions associated with servicing read requests. It turns out that such an optimized platform can result in performance 10x times more efficient than a traditional CPU server platform.

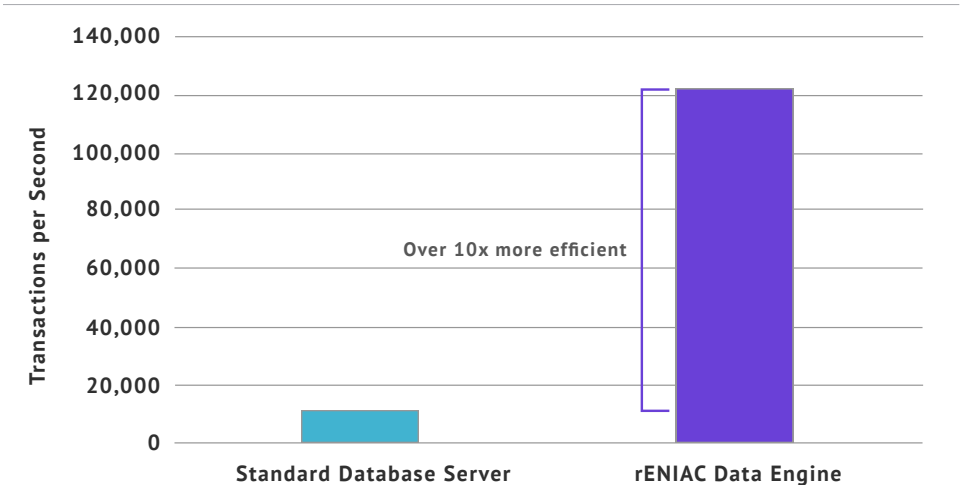


Figure 3. Data Throughput Comparison

A small number of rENIAC server nodes can handle a volume of requests that may have required hundreds of standard database nodes to handle; all while delivering dramatically lower and more deterministic latency. Built on a proxy-based architecture, rENIAC can be deployed with no changes to existing database software or application architecture and provide immediate performance improvements.

rENIAC Data Engine deployed as data proxy for scaling Cassandra read performance

A transparent proxy is one method to deploy rENIAC Data Engine that allows for immediate acceleration of existing database clusters with no changes to existing software or application code. We will focus on this deployment method for the majority of the paper, but additional deployment models are listed below.

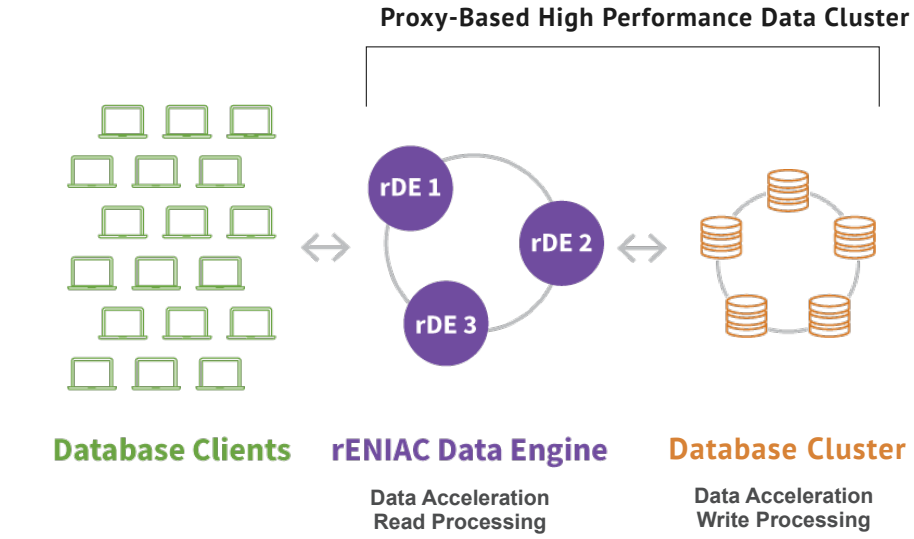


Figure 4. High Level Overview

A high-performance database cluster can be created via a proxy-based architecture, with standard database nodes front-ended by a small number of rENIAC Data Engine (rDE) nodes that caches data in terabytes per node using SSDs. In a proxy-based architecture, reads are serviced by the optimized rDE node, writes and other background management tasks are passed to the back-end database all without requiring any changes to application software.

The results can be astounding. When a Data Engine or Engines are added to an existing cluster, two things immediately occur:

1. Users experience predictably low latency, even at high loads.
2. The CPU utilization of the back-end database nodes reduces from 4x-8x, which in turn leads to faster writes, faster compaction, and faster compression.

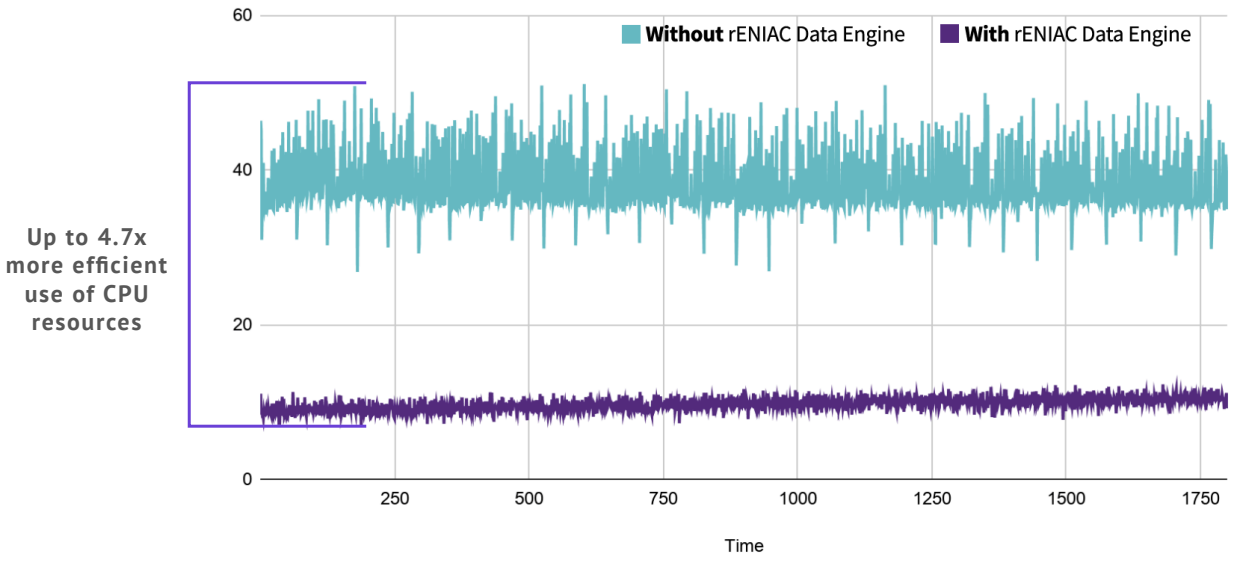
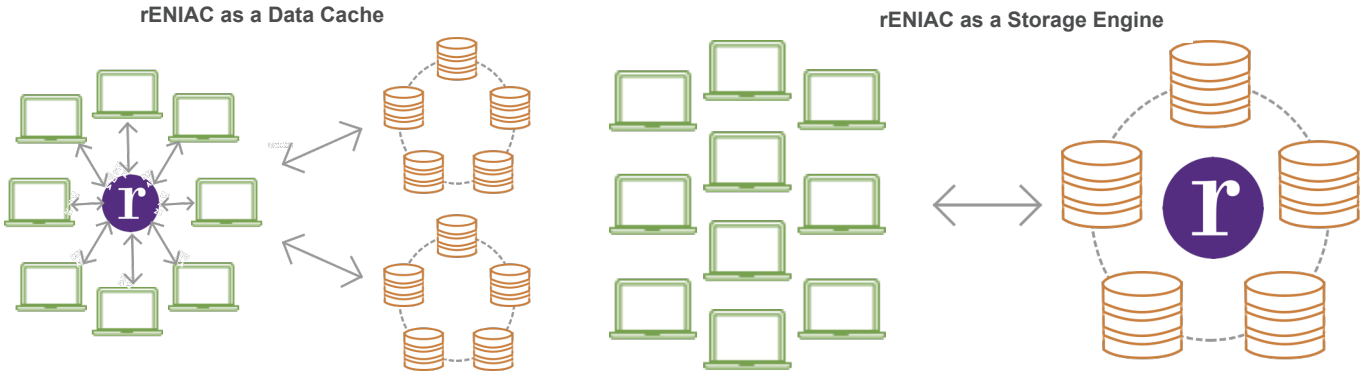


Figure 5. The Data Engine nodes free up database node resources so they can perform their essential tasks faster and more often.

The advantage related to offloading the back-end database nodes can be leveraged in a variety of beneficial ways:

- The treadmill of continually adding nodes and re-partitioning data can be halted.
- Back-end services such as compaction, analytics, reporting, and back-up can be scheduled more frequently with no impact on user performance.
- Individual node scale and cost and can be reduced.
- Overall node count can be reduced to serve the same volume of transactions and data capacity, thus reducing overall cost and complexity.

The two other methods for deploying rENIAC Data Engine that provide substantial efficiency gains for CPU usage, performance and latency are:



rENIAC as a Data Cache provides:

- Managed data access and retrieval at the client level for higher performance and flexibility
- Application or client server level updates to eliminate consistency or replication errors
- Multi TB data sets of access on a single node

rENIAC as a Storage Engine provides:

- Offloading of operationally-intensive workloads like data flushing, that normally hits the disk
- Write acceleration through compaction and automated key-value-store-like write automation
- Decrease in CPU usage and disk hits

Summary: The benefits of a finely tuned database cluster

A rENIAC-enabled, finely tuned database architecture supports massive growth of data and users via a small and cost-effective footprint while delivering low latency at scale; all with no changes required to present databases. Huge performance, direct cost, space, power, and administrative overhead benefits can be achieved. Those benefits to the data architecture are:

The cluster is *supercharged*. Both read and write maximum capacity are immediately increased. Predictable sub-millisecond latency is achieved even at heavy load. Additional capacity can easily be achieved by adding another Data Engine. Compaction and other back-end functions are accelerated. SLAs are attainable under all conditions, including windows of peak or intense traffic.

The cluster is *right-sized*. If a user already has a large cluster, “right” might mean capping the node count for now and forever, adding new data onto existing nodes. For a new cluster, “right” might mean only deploying a few nodes as driven by the needs for peak write performance.

The cluster operations are *simplified*. No time, effort, and expertise are required any more to execute and continually re-execute data partitioning. Back-end operations don’t have to be scheduled for off-peak times. Far less expertise is needed in general to plan and manage the cluster.

The cluster is both past and *future-proofed*. No database modifications are needed. You don’t need to move away from your existing SQL or NoSQL database.

Leveraging the power of FPGAs

Performance scaling in CPUs has been curtailed for many years now. Achieving 2x performance every 18 months just by switching to the latest and greatest CPU isn’t an option anymore. FPGAs, however, have been riding the Moore’s law while escaping the effects of breakdown of Dennard scaling. In addition, most FPGA designs tend to rely on spatial computing architecture utilizing micro-architectures that use custom memory architecture with dedicated interconnection topologies at much lower frequency compared to conventional CPU-based architectures.

In the past, FPGAs have been used to great advantage to scale networking (e.g., network switches and routers) as well as Flash controllers. The Internet would not be the Internet of today without hardware acceleration that has overcome the limitations of standard CPU architectures.

Modern FPGAs with much larger capacities enable augmentation of data acceleration functions such as storage I/O and network I/O, creating an efficient implementation architecture for data platforms such as database, search, and analytics.

rENIAC Data Engine (rDE) is available in a version that leverages FPGAs for both on premises and cloud-based deployments. For efficient read operation of database, rDE is deployed as a proxy between a database client and database

WE HAVE SEEN THIS BEFORE

The concept of using hardware assist to perform repetitive tasks for which CPUs are not optimal is not a new one. There are many examples of how such architectural function partitioning has been leveraged in the past. You use one every day — the Internet.

Back in the late 1980s, LANs struggled to run at 4-5 Mbps and WAN connections were considered fast if they ran at 1.5 Mbps. At that time, all packet processing functions were executed by software running on standard CPUs. Then the FPGA, ASIC and network processor revolution started and now 30 years later local area networks run at >1 Gbps, and 10 Gbps and 100 Gbps WAN pipes are common.

Hardware assist can change the world.

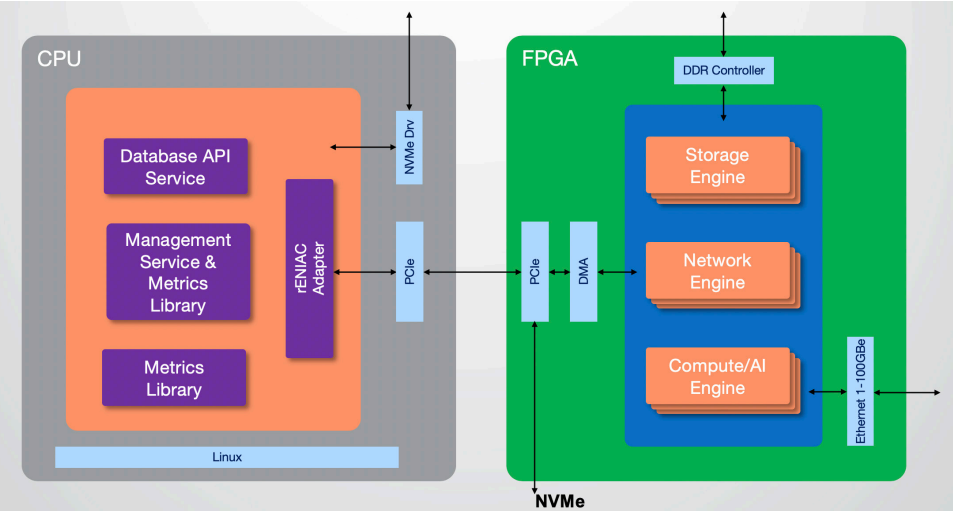


Figure 6. High Level Product Architecture

node, caching the data in (Flash) storage that is accessible by the FPGA hardware that has been programmed to execute repetitive functions in hardware for which CPUs are highly inefficient. It responds to queries by serving data either from its local storage or fetching it from the back-end database when the data does not exist in the local storage. This ensures that read requests are serviced with predictable sub-second latency and allows rDE to achieve throughputs much higher than a standard database cluster.

HOW DOES THE DATA ENGINE WORK?

The figure below shows a high level architecture of the FPGA-based rENIAC Data Engine. It is a hardware-software architecture involving components running on CPU as well as components relying on FPGA acceleration. The Data Engine is designed to support multiple databases and solve various data operations using hardware-software optimization techniques.

The main components of rDE are described below:

Storage Engine — rENIAC's Storage Engine is a key-value store in FPGA with a C/C++ API in host software. The storage engine is data model aware. It reads the schema and maps the data model onto the key-value store to mimic a columnar storage layout (as found in NoSQL databases).

Network Engine — rENIAC's network engine is a TCP/IP acceleration engine that supports TCP, IP, ARP, DHCP, and ICMP protocols implemented in FPGA and is interoperable with Linux in client and server configurations. This also implements acceleration of clustering specific to a database.

Compute Engine — Compute engines are a collection of acceleration engines implemented in FPGA that can be enabled for each instance of rDE to enable additional computational features such as Compression, Decompression, Encryption and Decryption. An option to connect to off-the-shelf AI inference engine also exists.

rENIAC's Data Engine provides FPGA acceleration of data platforms without DB or DevOps engineers needing to know about programming FPGAs. It provides greatly enhanced performance with zero. The result is reduced server node creep, reduced CAPEX and OPEX, sub-millisecond latency, and predictable SLAs.

WHY INTEL FPGAS

FPGA (Field Programmable Gate Arrays) allow a blank slate to build the solution that fits the problem best instead of fitting a solution into a predefined architecture. FPGAs provide flexibility for system architects searching for competitive accelerators that also support differentiating customization. Due to their configurability and hardware processing, FPGAs enable parallel processing as well as low latency, deterministic responses. FPGAs have been recognized for their increasing role in solving performance and analytical problems in data centers.

The Intel® Programmable Accelerator Card (PAC) features an Intel® Arria® 10 GX FPGA, an industry-leading programmable logic built on 20 nm process technology, integrating a rich feature set of embedded peripherals, embedded high-speed transceivers, hard memory controllers and IP protocol controllers. Variable-precision digital signal processing (DSP) blocks integrated with hardened floating point (IEEE 754-compliant) enable Intel Arria 10 FPGAs to deliver floating point performance of up to 1.5 TFLOPS. Arria® 10 FPGAs have a comprehensive set of power-saving features. Combined, these features allow developers to build a versatile set of acceleration solutions.

This PAC acceleration card for data centers offers both inline and lookaside acceleration. It provides the performance and versatility of FPGA acceleration. The card can be deployed in a variety of Dell EMC PowerEdge servers with its 1/2 length, single-slot form factor, low-power dissipation, and passive heat sink. The Intel PAC with Intel Arria 10 GX FPGA can be implemented in many market segments, such as big data analytics, artificial intelligence, genomics, video transcoding, cybersecurity, and financial trading. To help protect systems from FPGA-hosted security exploits, the Intel PAC with Intel Arria 10 GX FPGA features a Root-of-Trust device that enables more secure loading of authorized workloads and board updates and enforces policies to help prevent unauthorized access to critical board interfaces and flash memory.

The Intel PAC with Intel Arria 10 GX FPGA is one of several platforms supported by the Acceleration Stack for Intel® Xeon® CPUs with FPGAs. This acceleration stack provides a common developer interface for both application and accelerator function developers, and includes drivers, application programming interfaces (APIs), and an FPGA interface manager. Together with acceleration libraries and development tools, the acceleration stack saves developer's time and enables code re-use across multiple Intel FPGA platforms.

The Intel Acceleration Stack for Intel Xeon CPU with FPGAs is a robust collection of software, firmware, and tools designed and distributed by Intel to make it easier to develop and deploy Intel FPGAs for workload optimization in the data center. The Intel Acceleration Stack for Intel Xeon CPU with FPGAs provides multiple benefits to design engineers, such as saving time, enabling code re-use, and enabling the first common developer interface.

FPGAs are an essential technology to meet demanding high-performance requirements, but software is needed to take advantage of their benefits without changing existing applications. rENIAC leverages the power and flexibility of Intel FPGA platforms in Dell EMC PowerEdge servers by providing software solutions that override (i.e., bypass) traditional CPU data flow to accelerate data and traffic without requiring changes to the application software. The result is predictable workload performance for private and public clouds running critical NoSQL workloads — without requiring knowledge on designing with or programming FPGAs.

Real-time use cases for rENIAC Data Engine

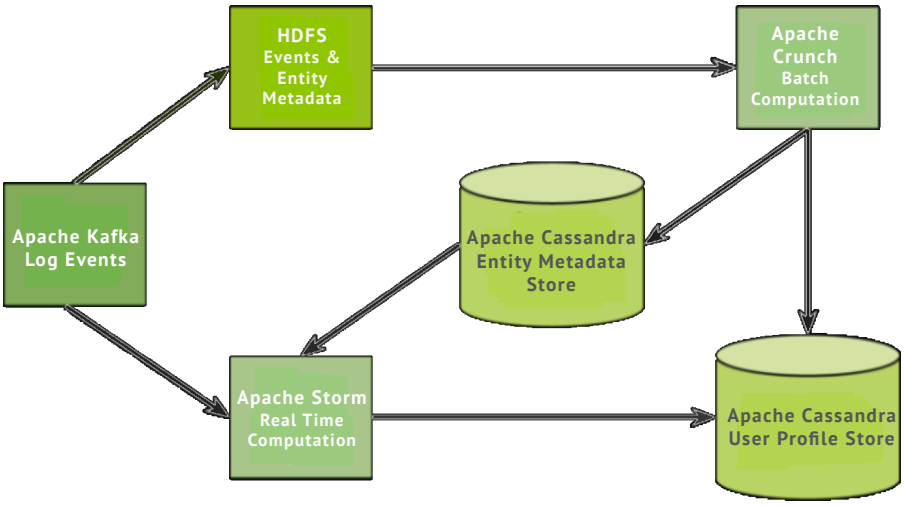


Figure 7. User Profile Store

USER PROFILE STORES

Cassandra, along with other databases, is often used in “Personalization Stacks”[1], where user information must be looked up and analyzed quickly in order to take action at click rates, as a user connects to or browses a site.

For popular ecommerce and financial sites with large numbers of users, the amount of users, records, and stored data can be very large, and continually grow as the user base expands. But performance remains critical because if users can’t log in quickly, they rapidly move on to other sites.

This is an ideal use case for rENIAC Data Engines to create a high performance architecture that maintains low latency and stabilizes back-end costs even as data continually grows.

1 User Profile Reference Architecture, Spotify blog “Personalization at Spotify Using Cassandra.”

RECOMMENDATION ENGINES

Similar to the User Profile Store use case above, specific user information, which can be nuanced and based on continually increasing amounts of data, must be accessed and analyzed quickly in order to provide appropriate and valuable recommendations to that user. Low latency to ensure user satisfaction is a critical requirement while at the same time minimizing and stabilizing back-end costs. Typically found in ecommerce sites, this is another ideal use case for the rENIAC Data Engine.

PRODUCT CATALOGS

Responding with relevant and exact search results at the time of customer query plays a huge part of online shopping customer experience alongside inventory and availability, checkout and payment processing.

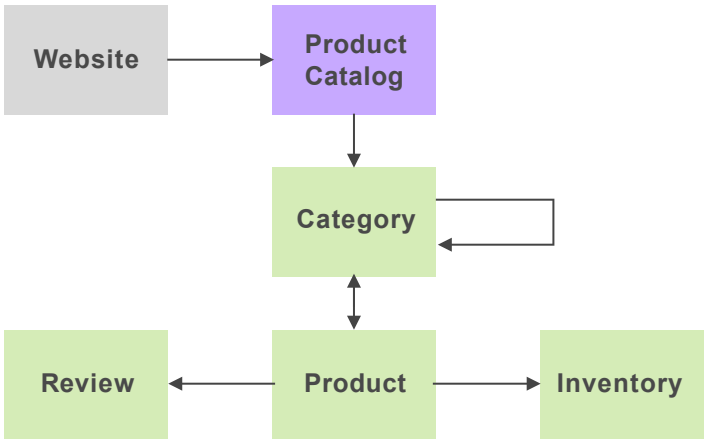


Figure 8. Product Catalog

By accelerating product catalog results to instant real-time levels and pairing those results with data points like location, user profile and recommendation engines, ecommerce and retail businesses can increase cart size and revenue, which makes it an ideal use case for rENIAC Data Engine.

FRAUD PREVENTION

Fraud prevention comes in many forms. One, related to credit card transactions, looks at every transaction and compares it to past transactions from a specific consumer based on parameters such as store, geography, amount, frequency, etc.

Scale is imperative for this use case as credit card companies have many users, and the more past transactions that are analyzed, the better the fraud prevention. A high performing database like Cassandra combined with the rENIAC Data Engine meets the needs for this use case.[2]

2 Fraud Detection and Prevention Reference Architecture, adapted from the OpenCredo blog "Data Analytics Using Cassandra and Spark."

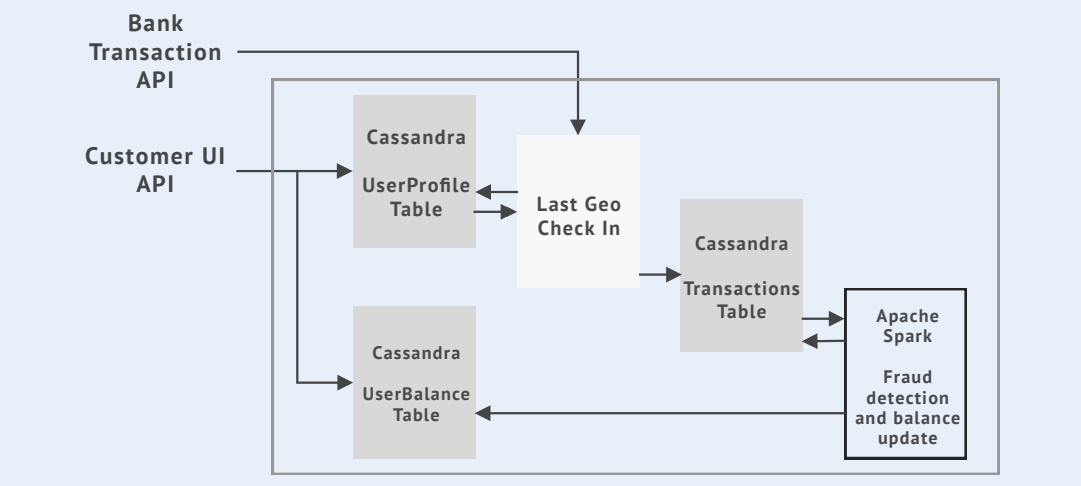


Figure 9. Fraud Prevention

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Artificial intelligence and machine learning are excellent use cases for rENIAC Data Engine as they often require rapid computation on large amounts of data to meet user expectations. Whenever a NoSQL or SQL database is involved in the storage, management and real-time access of the operational data, rDE can be implemented to improve read data access by 10x while also increasing hot data storage by an order of magnitude to multiple terabytes. If online or real-time machine learning is required to improve recommendation engines or personalization, for example, both processing capability and large volumes of data need to be at the ready. Such situations occur widely, in finance, gaming, healthcare, energy, ecommerce and retail, autonomous or assisting driving, manufacturing and IoT.

Reference benchmarking of rENIAC Data Engine with Cassandra NoSQL

Node type	3x Database Server	3x RDE Server	2x Client Server
Chassis	2U Dell PowerEdge R740		
CPU	Dual Socket Intel® Xeon® Gold 6130 (16-core) CPU @ 2.10GHz – Hyper threading enabled		Dual Socket Intel® Xeon® Silver 4114 (10-core) CPU @ 2.2 GHz – Hyper threading enabled
Memory	64 GB (4x 16GB 2666 MHz DDR4)		
Storage	Samsung SSD 970 EVO Plus 1TB		
Discrete FPGA		1x Intel Programmable Acceleration Card (PAC) with Arria 10 FPGA PCIe card	
Intel Acceleration Stack		**Quartus**: *Version 17.1.1 Build 273 12/19/2017 Patches 1.01dcp,1.02dcp,1.36,1.38 SJ Pro* **aocl:** *17.1.1 Build 273* **DCP1.2pv**: *a10_gx_pac_ias_1_2_pv*	
SW Requirements	**OS:** CentOS 7.4.1708, **Python:** 2.7.5, **Apache Cassandra:** 3.11.4, **CQL:** 3.4.4		

The configuration for benchmark data is 2-3-3 [2 client nodes, 3x rDE nodes, 3x servers], Dell EMC PowerEdge servers connected with a PowerConnect switch, with 28 application threads and 4k payloads, running Cassandra-stress.

The benchmark data was obtained by comparing baseline performance of Apache Cassandra 3.11.4 against rENIAC deployed as a cache (100:0, read:write) and with rENIAC acting as a data proxy (80:20, read:write) with Cassandra as the persistent database.

PERFORMANCE METRICS

We observed two major improvements in rENIAC's performance over baseline, specifically throughput per second (TPS) and latency. The full performance results are shown in the two tables below.

Table 1. 100:0 (read:write) benchmark results — rENIAC Data Engine (rDE) deployed as a cache - 4k payloads

	Baseline Cassandra	rENIAC Data Engine	GAIN
Operating rate (Ops/sec)	12,562	133,915	***10.7x***
Latency mean (ms)	4.4	0.4	11.0x
Latency median (ms)	3.4	0.4	8.5x
p95 latency (ms)	9.2	0.5	***18.4x***
p99 latency (ms)	14.5	0.6	24.2x
p99.9 latency (ms)	166.0	1.1	150.9x
Latency max (ms)	1097.9	29.9	36.7x

When rENIAC is serving all read requests and acting as a highly optimized cache, the following performance improvements can be realized:

- 10x (or higher) throughput, i.e., operations per second — in this test, rDE was 10.66x more performant than Cassandra alone at serving read requests
- Consistent sub ms latency — in this test, rDE was serving reads at sub ms latencies, only crossing over the 1ms mark at p99.9 latency
- 18x (or greater) p95 latency gain — in this test, rDE was 18.4x faster at serving requests at p95

Table 2. 80:20 (read:write) benchmark results — rENIAC deployed as a proxy - 4k payloads

	Baseline Cassandra	rENIAC Data Engine	GAIN
Total TPS	14,752	62,823	***4.3x***
Ave p95	9.0	2.2	4.1x
mean (ms)	3.8	0.9	4.2x
p95 (ms)	17.9	4.3	***4.1x***
p99 (ms)	28.8	14.0	2.1x
p99.9 (ms)	350.8	101.7	3.4x
max (ms)	2580.5	1417.7	1.8x

When rENIAC is acting as a proxy for mixed workloads, the following performance improvements can be realized:

- 4x (or higher) throughput, i.e., operations per second — in this test, rDE was 4.3x more performant than Cassandra alone at serving read and write requests
- Mean sub ms latency — in this test, rDEs mean latency is 0.85ms, a 4.2x improvement over baseline Cassandra
- 4x (or greater) p95 latency gain — in this test, rDE was 4.1x faster at serving requests at p95

Results summary

When addressing the read inefficiency in an open source database, whether deployed as a proxy, a cache, or both, rENIAC provides significant performance improvements in the form of throughput (up to 10.66x) and latency (up to 15x) when measured against the database alone, like Cassandra. These improvements provide users with the following benefits:

- **Consistent SLA achievement** — predictable sub ms latency means fewer timeouts, spikes and satisfied end users as SLAs are consistently met.
- **Operationalization of data in real time** — higher throughput and lower latency means larger data sets can be operationalized in real time instead of waiting to process offline. This gives ML models rapid learning cycles and provides end users with accurate and personalized recommendations instantly, which increases revenue.
- **Increased headroom or reduction in cost and complexity** — getting 10x more out of existing infrastructure means greater CPU headroom, which can be used to add features or create additional revenue-generating applications. If cost reduction is a goal, these performance improvements enable customers to maintain existing infrastructure and achieve business and technical goals without continuously adding new servers or hardware.

To learn more, visit delltechnologies.com/referencearchitectures