rENIAC

Data Store Acceleration-as-a-Service on Amazon FPGA Instances

Prasanna Sundararajan, CEO, prasanna@reniac.com
Chidamber Kulkarni, CTO, chidamber@reniac.com

This paper describes the use of rENIAC's Distributed Data Engine on FPGA-based AWS F1 instance to deliver higher throughput and lower latency within a scale-out architecture. The Distributed Data Engine is architected to address bottlenecks involved in any data infrastructure workloads. Our case study shows 10x reduction in latency and 9x increase in throughput with no software change for an existing Cassandra Database customer resulting in significant value-add to end customer business. rENIAC's unique approach to deploy F1 instance as a network service eliminates the need to re-architect or recompile application code.

rENIAC, Inc.
1621 W El Camino Real, Suite 102
Mountain View, CA 94040 USA
contact@reniac.com
www.reniac.com

 Version: 1.0

Data Store Acceleration-as-a-Service on Amazon FPGA Instances

In data-intensive businesses, scale and speed is money

The Challenge of Data-driven Business

To grow their top line and drive large cost-savings, players across sectors including banking, retail, digital media, and others must overcome a major challenge: processing large volumes of data to personalize product offerings for consumers. Consider one simple but dramatic example: retail giant Walmart, Inc., which services millions of customers at any given moment, has shown that a one-second reduction in service level agreement (SLA) time to target the right product to an online customer can increase conversion rate by as much as 2%, driving large revenue improvement.

The competitiveness of online businesses and enterprises depends on their differentiated technology infrastructure.

Data Workloads, Limitations of Scale-out Architecture & Performance Engineering

Data Workloads today are dominated by
- Application: Map/Reduce, Spark or custom
- Data stores: Hadoop FS, NoSQL, SQL

Our analysis has shown that up to 75% of CPU cycles is spent on Input/output operations resulting in bottlenecks. These bottlenecks inhibit scaling and meeting SLA in some instances. While in other cases customers have to pay more to meet their desired performance goals or compromise on performance.

These workloads and data stores run on commodity servers. When transaction or data volume increases, more servers are added, as part of a scale-out architecture approach. That's an inadequate solution to the I/O bottleneck problem. With the adoption of scale-out architecture, bottlenecks present in existing databases still limit their ability to scale linearly, especially when cluster size increases beyond a certain limit. Here are examples of problem statements from customers:

Customer in need of performance: "Even after upgrading servers with larger DRAM and SSDs and horizontally scaling for IOPS, we are unable to meet 95th percentile SLA for DB transactions."

Customer in need of smaller footprint: "While scaling for IOPS at a particular CPU % utilization to meet our SLA, we had to account for compaction and repairs, which forced us to over provision and double our NoSQL cluster size."

Customers are forced to
- Spend more (add more nodes than planned), or
- Suffer reduced capability (due to performance compromises)

Existing options to optimize for performance comes at the cost of the application developer's productivity. Developers are forced to recompile and re-architect their code to make the most of what's offered by the hardware (for instance, a multi-core CPU), or port their code to new platforms such as ones based on GPUs, and FPGAs. This requires augmenting a customer's organization with developers with skills and expertise in these new platforms.

Amazon F1 Instance

With the wide-spread use of commodity servers for scale-out architecture, the use of proprietary vendor-built appliances that tightly integrate software and hardware is no longer an option. However, FPGA-based solution can be used for tight software and hardware integration to deliver the performance and optimization of an appliance. With the introduction of FPGA-based instances in AWS, customers can derive the benefits of the appliance within the scale-out cloud architecture.

Sources of indeterminism in traditional CPU-based architecture such as cache misses, context switches etc. are avoided when mapping an application to FPGA-based architecture. Thus a customer is able to get deterministic performance. And given application is directly mapped to FPGA hardware, one gets lower latency per transaction that is not possible with software running on CPU-based systems.

From a development standpoint, proving and testing the scalability of solution for large clusters was only possibly by setting up on-premise clusters which requires extensive investment. However, with availability of F1 instance, on-demand testing of FPGA-based solutions is a viable option to test and prove the scalability of application of our solution using large clusters.

In the next two sections, we will discuss rENIAC's unique approach to enable network architecture on F1 instance. This network architecture is foundational part of rENIAC's Data Engine that is designed to address various IO bottlenecks in data workloads executed on a scale-out architecture.

rENIAC's Unique Network Architecture on F1 instance to Enable as-a-Network-Service Deployment

Figure 1 shows a high-level view of the AWS F1 instance architecture. The key point to note here is that the network interface is connected to the CPU. Hence the accelerated function on FPGA needs to be fed data from the CPU. This is very much like a co-processor model for building accelerators. However, one of the key strengths of FPGAs are in applications that require low-latency response. This is possible with FPGAs since they tend to be very good at accelerating networking functions such as TCP termination and protocol processing (e.g. HTTP, SSL, etc).

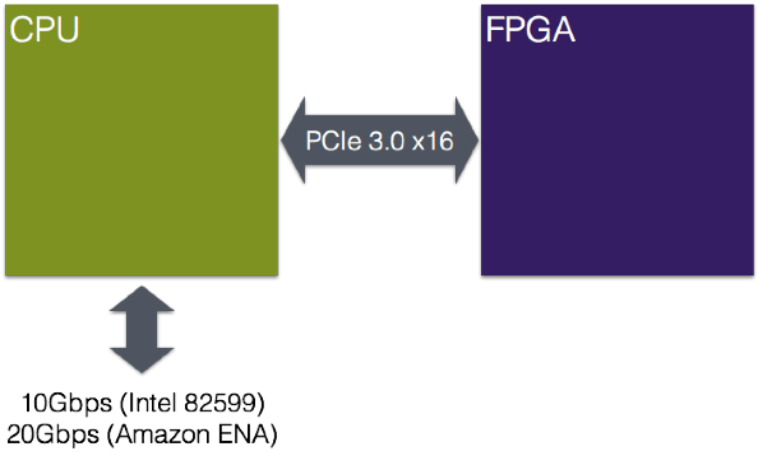


Figure 1: High-level view of AWS F1 Instance

Today the most common approach to accelerating applications on the F1 instance is to terminate TCP on the CPU and then feed the application data to the FPGA. This will result in additional overhead both in latency, since the network processing has to be done in CPU now, as well as need for additional CPU cores (e.g. TCP termination at 10Gbps on a Intel Xeon E5 class CPU will require at least 3-4 dedicated cores). This will result in either a need for CPU instances with more vCPUs or a more complex application partitioning between CPU and FPGA resulting in longer development time.

rENIAC's network stack for F1 instance enables customers to bypass the Linux kernel and transport raw ethernet packets to the FPGA directly. rENIAC's production tested network engine enables customers to terminate TCP in the FPGA while exposing a standard socket API for connecting to applications.

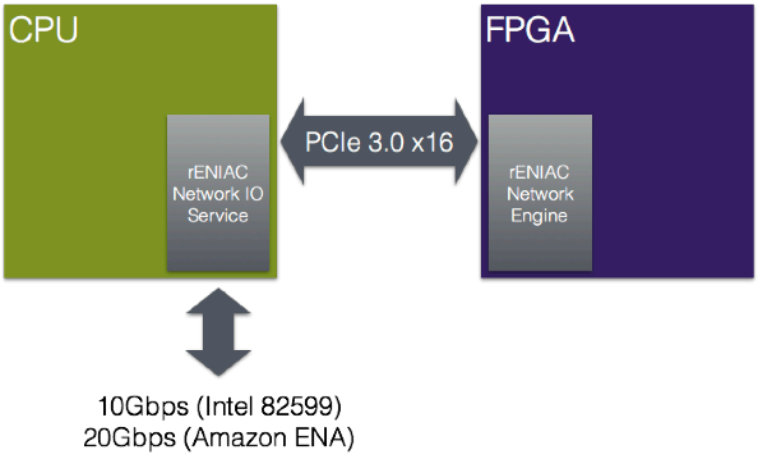


Figure 2: High-level view of AWS F1 Instance with rENIAC Network IO service and Network Engine

Figure 2 shows a high-level diagram of this stack. There are two key blocks in our network stack. First, a network IO services block in software that is used to bypass the Linux kernel and provide a low latency interface between either Intel 82599 or Amazon ENA network adapters and PCIe interface to FPGA. Second, a network engine on FPGA that implements full server-side and client-side TCP functions that are needed to co-exist with a Linux-based data center environment. Our network engine is production tested and has withstood traffic for 45+ days in production environment.

This network stack is key building block to deploy rENIAC's Distributed Data Engine as-a-network service.

Distributed Data Engine

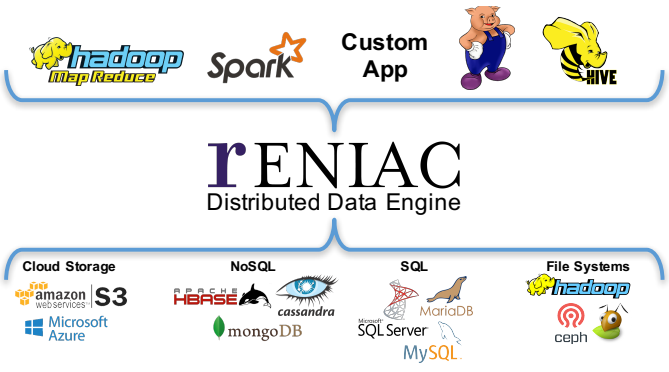
The Distributed Data Engine solves the IO bottleneck problem cost-effectively. It's designed to address key inefficiencies when executing data applications, providing higher performance at lower cost on commodity servers, but without required software changes while supporting operational features (e.g., business continuity, and performance monitoring).

The Engine does this by enabling decoupling of the data and application layers, simultaneously acting as an IO accelerator to resolve any bottlenecks. Large volumes of transactions are processed by the engine's unique proprietary architecture that includes:

- **Query Processor** that is network attached to speed up transaction processing
- **Tiered Storage Class Memory** coupled to the network stack for predictable low query latency
- **Inter-node Router** to accelerate the transfer of data and control traffic between the nodes
- **Compute Engine** to speed up various operations (e.g. filters, scans, compression etc.)

The engine's runtime plug-in software enables seamless integration with any existing DBs and ensures no application code change is required while respecting the consistency protocols of the underlying data stores.
From ease of deployment of the Data Engine, it is designed such that an Ops engineer is required only to edit the configuration file to point the target IP address of the service. Since the Data Engine expects application servers to send Application level commands directly to the FPGA, having a low latency network stack is essential both from performance and as-a-service deployment standpoint

Throughput	Up to 40x increase
Latency	1/3 -1/10 lower predictable latency
Deployment	Datastore as a network service for on-premise, cloud or hybrid (mix of on-premise & cloud)
Technology	Uses state of the art technology: FPGA, CPU, Memory, SSD Intel 3D XPoint technology ready
Operational features	Redundancy & Fail-over for increased business continuity, metrics for operational visibility
DataStores	NoSQL, SQL & File Systems. Runtime API plug-in ensures no SW changes.

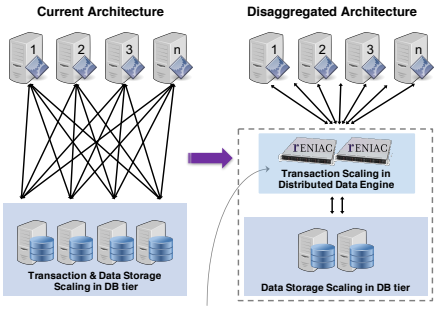


NoSQL Case Study from Leading Digital Media Company

A large, online digital media company that relies heavily on personalized content was storing user personalization data in Cassandra. They were only able to meet their performance targets at 75th-percentile service level agreements (SLAs) for database (DB) transactions, resulting in unrealized revenue. In

addition, they faced 30% year-over-year growth in number of DB servers, to support current business growth.
Current solutions forced them to scale their Cassandra nodes both for transactions and data storage. This significantly increased their infrastructure footprint and cost structure, without improving SLAs.

rENIAC's Distributed Data Engine (as a network service), enables disaggregation of transaction and data storage scaling. The result is a 9x increase in transaction throughput and 10x reduction in latency (5-8 msec SLA at 99th percentile), improving speed and revenue while lowering cost.



rENIAC tech. deployed as Database-as-a-Network-Service

	Current Infrastructure	With rENIAC	rENIAC Advantage
DB queries per node #	2,905	20,000-26,000	60%lower TCO
Latency per query (i.e.SLA) P = percentile	75th P: 7-8ms 95th P: 35ms 98th P:60ms	99th P: 5-8ms	Increased Revenue
Software API	Cassandra Community 2.1.13	Cassandra Community 2.1.13	No changes needed

Value: 9x increase in throughput, 10x lower latency

Applicability of Xilinx High Level Synthesis Tools to Validate and Prove Product Features without Hardware Deployment

rENIAC has been an early adopter of Xilinx High Level Synthesis (HLS) technology. rENIAC benefitted in two ways by using HLS technology - first, conquer complexity of FPGA development and second, provide customers with software emulators early in development cycle to validate product features.

Use of Xilinx HLS technology in our development process gave us two main advantages. First, it has allowed us to conquer complexity by focusing on function rather than implementation details. The fusion of implementation details with functional correctness for FPGA designs results in longer design cycles. Second, we were able to leverage large amount of software infrastructure in both data and networking world for verification of our 'C' specification. This helped us to rapidly get a working design.

Finally, the most important advantage we believe HLS technology provided us was the ability to ship a software emulator of our product to select customers much earlier in our development cycle. This allowed us to validate specific product features e.g. ease of integration of our product in to end customer stack with no change to their application software. For example, one of our customers had to only change the configuration setting in their yml file to point the target IP address to our software emulator to start using our stack. In a live data center environment, this is a very important feature and we benefitted immensely with using HLS.

About rENIAC

rENIAC's team (HQ in Mountain View, CA), is a group of passionate entrepreneurs with data, software and systems experience from companies such as LinkedIn, IBM, Xilinx, Samsung, Riverbed and Napatech. rENIAC has 3 patents awarded with more pending. rENIAC technology is production deployed at Tier-1 media company. rENIAC is VC backed and guided by industry veterans from Virident, HP, Synopsys, Facebook and Hortonworks.