Method and apparatus to implement computational storage to provide an Infrastructure Platform

Joydeep Das, Kulwinder Atwal
Business Unit: HIT
Email Addresses: das@hpe.com, kulwinder.atwal@hpe.com

Abstract

This document describe the method and apparatus to implement a distributed cluster of containers within a chassis using computational storage with the help of server virtualization techniques, VNIC and RDMA where each drive serves as a compute and storage node and runs Linux containers on each storage drive to overcome the processing bottleneck inherent CPU designs, by offloading work, increasing the core count, and reducing data latency while lowering power consumption, physical size, and cost.

Problem statement

In today's distributed compute clusters, both compute latency and network latency play a major role in the data processing bottleneck. While a distributed compute cluster eliminates the need of costly high core-count CPUs, latency increases as more nodes come into the cluster. Additionally, due to the long data path from storage to the CPU core, many structures have been developed in the CPU to mitigate the delay such as three levels of cache, cache coherency logic, memory bus snooping by the cache controller, high-speed expensive DDR RAM DIMMs that improve throughput while increasing latency and consume power much more than the cores themselves. Also, even when an application is sized correctly for CPU cores/Memory/Storage there is still bus contention between CPU cache, memory, and PCIe local storage result in millisecond wait times before storage device latency is even considered. Applications need a higher core count and low latency access to their data to improve the processing bottleneck in a linearly scalable manner.

Our solution

Device: This solution use computational storage drives which come with their own CPU, memory, and local flash storage to create a Compute-Storage distributed cluster inside a single physical box. A single flash controller chip, in addition to flash controller logic, is comprised of 4 CPU cores to run Linux containers. Each 1 GHz core is speed matched with the DRAM, thus eliminating all the above cache related structures and expensive external memory DIMMs resulting in significant power and physical size savings. Each core is provided dedicated direct access to flash channels resulting in zero bus contention between the Core, RAM and local storage running a single container resulting in data path latency in the microseconds. Offloading data-centric tasks such as Erasure Coding, compression, encryption, deduplication, data analytics, and machine learning also helps with the processing bottleneck.

Architecture: The solution is matched with an AMD CPU on the motherboard that supports PCIe Peer-to-Peer traffic in its on-chip PCIe Root Complex. This feature enables a low latency data path that bypasses software on the AMD CPU, allowing Smart vNICs and Computational SSDs to scale independent of the CPU bottleneck. Smart NICs combined with Computational SSDs enable the promise of SSD-speed RDMA by making use of either Peer-to-Peer PCIe offered on AMD CPUs, or traditional SR-IOV offered on Intel CPUs by matching a PCIe virtual function on a NIC with a PCIe virtual function (a Linux Container) on an NVMe SSD using PCIe bus mastering.

Infrastructure: Smart NICs and Computational SSDs can still offload management traffic to the AMD/Intel CPU, turning the main CPU into a low-cost low core-count management core. For example, Kubernetes can run on the CPU to orchestrate Linux containers on the SSD cores turning the AMD CPU into an internal management server. Many cluster types can be supported in this Linux container model such as CEPH, serverless cloud infrastructure running an official AWS Lambda Function runtime [1], SSD-speed RDMA, and NVMe/TCP.

Security: Authentication security is provided at the hardware level by the BMC's use of a Trusted Platform Module that can authenticate an immutable authentication key embedded in each SSD computational storage chip, while

transactional security is provided by hardware accelerated SSL in the SSD, providing security in a Zero-Trust environment from hardware to firmware and up to the application stack. This bottom-to-top security approach prevents hardware of unknown providence from entering the supply chain.

Infrastructure Platform: We present just one-use case in depth, but the infrastructure solution is applicable to any compute cluster application. This treatment is not comprehensive due to the limited space.

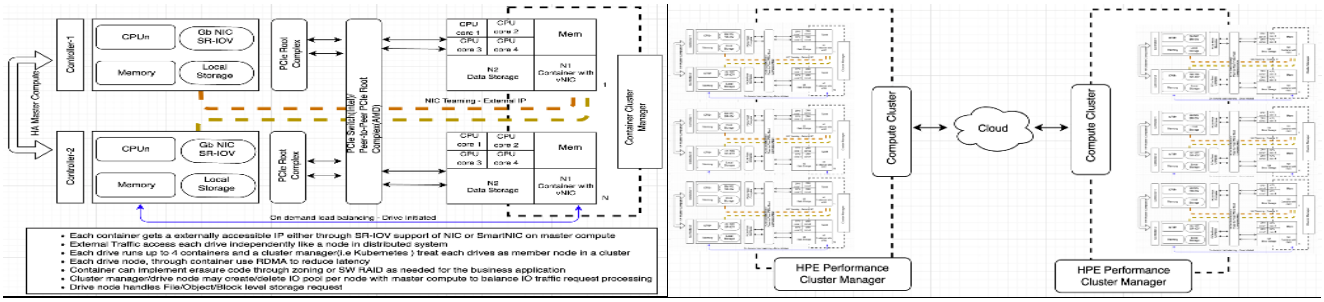
TECHCON 2020 Challenge 1: Secure Edge-to-cloud control plane

This high density, low-cost container platform is an ideal infrastructure solution for edge devices that need to generate alerts in a secure low-cost "serverless" solution. For example, an AWS specific solution would make use of Lambda Functions [1]. An AWS Lambda function is a small program that can algorithmically respond to triggers when a certain stimulus is presented. A Linux container is used to execute a Lambda Function created by the customer that is then transmitted securely using REST encrypted over HTTPS in a Multi-Tenant environment to the target. Kubernetes running on the AMD CPU is used to manage the creation, destruction, allocation, and access of the Linux containers that are located either locally or across the Internet. Since Lambda functions are transitory, the majority of the code can reside on the SSD flash. When an Event Handler in the container detects a request to the End Point the Lambda Function code is loaded, completed execution in microseconds, deallocated from RAM. Due to the low latency, Lambda Function code does not need to reside in the Computational Storage CPU RAM.

Hyper-scaler Agnostic Open API: Each Hyper-scaler can have its own specific container type deployed on demand by Kubernetes as the first phase. The second phase can create an HPE middleware in each container type to abstract out implementation specifics making customer work portable and Hyper-scaler agnostic.

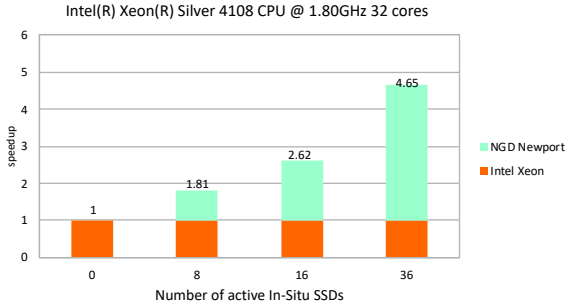
Analytics: Analytics engines powered by Machine Learning. Some SSDs containers could be tasked with Machine Learning inferences in Real-Time on data stored on its own SSD media. Each SSD analytics container can cooperate with other analytics containers using Peer-to-Peer PCIe to build a global view.

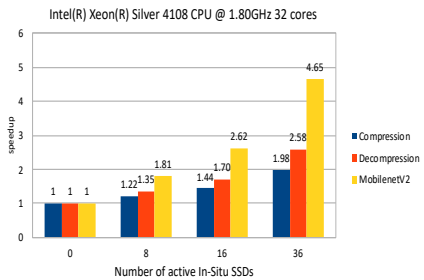
Fault Tolerance: Fault tolerance is provided by Kubernetes monitoring container health and starting new containers as needs in the cluster internal to the box. Two or more boxes provide Mastership election on failure of the Active box in a Hot/Standby High-Availability model, or an Active/Active High-Availability model.



Evidence the solution works

Performance: Here an Intel CPU with 32 cores augments its Machine Learning inferences for TensorFlow MobileNetV2 by offloading some work [2]. Data is normalized to 1, where 1 represents the amount of work done by the 32 core Intel CPU.

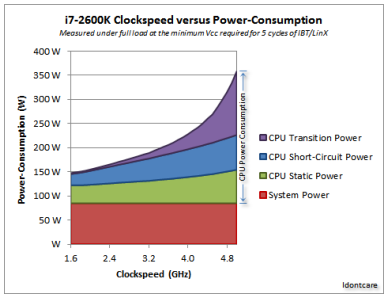


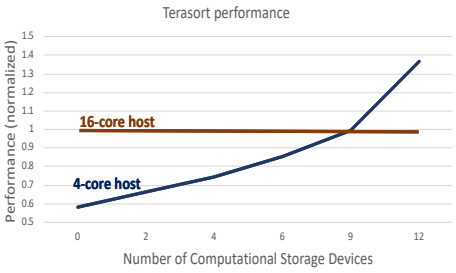


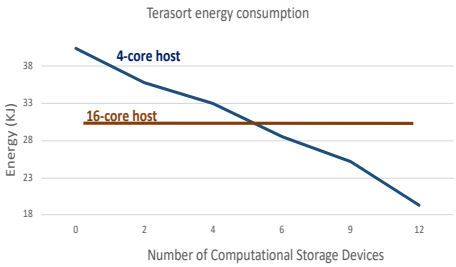
Adding Computational SSDs where the SSDs do all the work, starts to do better than an Intel Xeon CPU with 32 cores at about six to eight SSDs [2].

Size Reduction: Here we compare an 18RU full rack of nine 2RU servers each with an eight core Intel Xeon E5-2620v4 populated with 12 8TB SAS SSDs versus a 3RU rack space solution comprised of just three 1RU servers. Each server also has the same Intel Xeon E5-2620v4 main CPU, but also is populated with 36 EDSSF 8TB NVMe SSDs. Both solutions provide a total of 864TB of storage, but 432 cores are available for work in our solution versus 72 cores. With one Intel core equivalent to 4 cores in an SSD, we get approximately 108 equivalent Intel Xeon cores in a space that is 6 times denser [2]. Or, about 648 times more powerful in the same 18RU space.

Power Consumption: Power consumption can be stated linearly as Power = Voltage^2*Clock Frequency until 1 GHz. Then higher temperature causes more leak current, which in turn causes a higher temperature resulting in a non-linear increase in power consumption. This is known as the 4 GHz Power Wall.







By keeping the CPU clock speed at 1 GHz we can stay in linear power consumption. Teresort operation on a Hadoop cluster shows an Intel 16 core CPU may have the equivalent performance of nine SSDs, but the power consumption will be about 20% higher on an Intel CPU.

Competitive approaches

The approaches by ScaleFlux, Samsung, and Eideticom is to add an FPGA to the SSD, but does not present any processor cores to the application.

Current status

This solution is currently in its theoretical study state while working on various models in house, while performance data was provided by [2].

Next steps

Compile the open source AWS Lambda C++ Runtime and show serverless performance on Computational SSDs. Add a Python wrapper to allow current AWS customers using Lambda functions written in Python to run un-altered, thus reducing potential HPE Cloud customer stickiness to AWS while providing a lower cost per alert solution.

References

[1] AWS open source code for a “serverless” C++ Lambda Runtime https://github.com/awslabs/aws-lambda-cpp

[2] Computational Storage Performance.pptx – NGD Systems, Inc.