ScaleFlux

Guide 2024: *What is Computational Storage?*

And how to take advantage of it

A proven approach, now applied to storage

Key benefits

- **Offload tasks from the CPU to optimize resources**
- **Analyze more data by improving performance**
- **Improve efficiency by multiplying capacity**

Computational storage is not a new idea, particularly its new methodology.

You might be familiar with the idea of **graphic processing units (GPUs)** from NVIDIA or AMD and others that can perform some very specialized mathematics to relieve the CPU from the pressure of performing those functions.

These specialized processors can perform these functions even better than the general-purpose CPU. Pushing these tasks to a specialized processor lets users get more out of their CPUs. The industry had the idea of **Smart-NICs** or **Data Processing Units (DPU)**. For example, network interface cards with some compute resources on-board can do things like compression, decompression, encryption, and many other things at wire speed.

That brings us to **computational storage**. Simply put, it is the idea of a storage device with an element of **compute to perform a specialized function**. These can come in many different shapes and sizes. We now have **SSDs** with compute functions, devices on the memory bus, and add-in cards with specialized storage processors. They may have FPGAs or ASICs, but their primary role remains - **handle data-centric tasks for the CPU**. We use the right compute type of resource as close to data as possible to relieve pressure on the CPU and get more work done.

Let's look at the ScaleFlux approach

We take our SSD designed to be as performant and feature-rich as any major vendor of SSD devices. Then, we add our custom silicon to give it some additional functions that enable it to achieve better **performance**, better **capacity** utilization, and more **endurance**.

There are a couple of ways you could do this - with an **FPGA** or with an **ASIC** or **SoC (System-on-Chip)**. FPGAs are a bit easier to design, but they consume host resources and can be more complex in other ways. They're also not as fast or as power-efficient as an ASIC. While delivering high-performance ASIC with multiple functions is an extraordinarily tricky task (and very expensive if you get it wrong!), ScaleFlux has done exactly that with our first-ever Silicon design. This SoC sits at the heart of the **CSD 3000**, allowing us to **transparently compress data before writing to the NAND and decompress the data when reading it back from it.**

What does this deliver?

Very simply, if you write fewer bits to the NAND, you have an increase in performance from an IOPS or bandwidth perspective from the host. You can also run the device to a far higher capacity utilization before you see any abnormal performance side effects. So looking at the design ahead of you, you can see that it has a PCIe **Gen 4 by 4 interface**. We have our own customized ASIC on board that is then backed up by industry-leading NAND to persist and store the data.

It allows you to store more data - either by increasing utilization rates to 85-90% or by expanding the namespace beyond the physical capacity.

When we bring this all together, even with a modest amount of data compression, you get industry-leading performance in IOPS and bandwidth. You may see **double or even triple the capacity** you can use on the device because of the built-in data compression.

Finally, because we're writing less data to the NAND, you can go from a 5-drive write-per-day device to a 9-plus drive write-per-day device.

If you're writing less data, it means that your endurance and your reliability significantly increase.

Deep dive into the ScaleFlux approach

When designing a computational

storage device, first, we need to build an SSD that is physically equal to any other enterprise-class SSDs. So, in the design, we of course have some NAND. We use 16 channels in our ASIC to access that NAND. This is a common design practice for enterprise class SSDs. The drive is a U.2 form factor, one of the most common form factors for SSDs. As we move forward into **Gen-5**, we'll do other form factors such as **E1.S** and **E3.S**. We also need some DRAM that acts as a buffer area to land data before we commit it to the NAND channels, and it allows us to manage our flash translation layer while doing a few other functions with our DRAM as well.

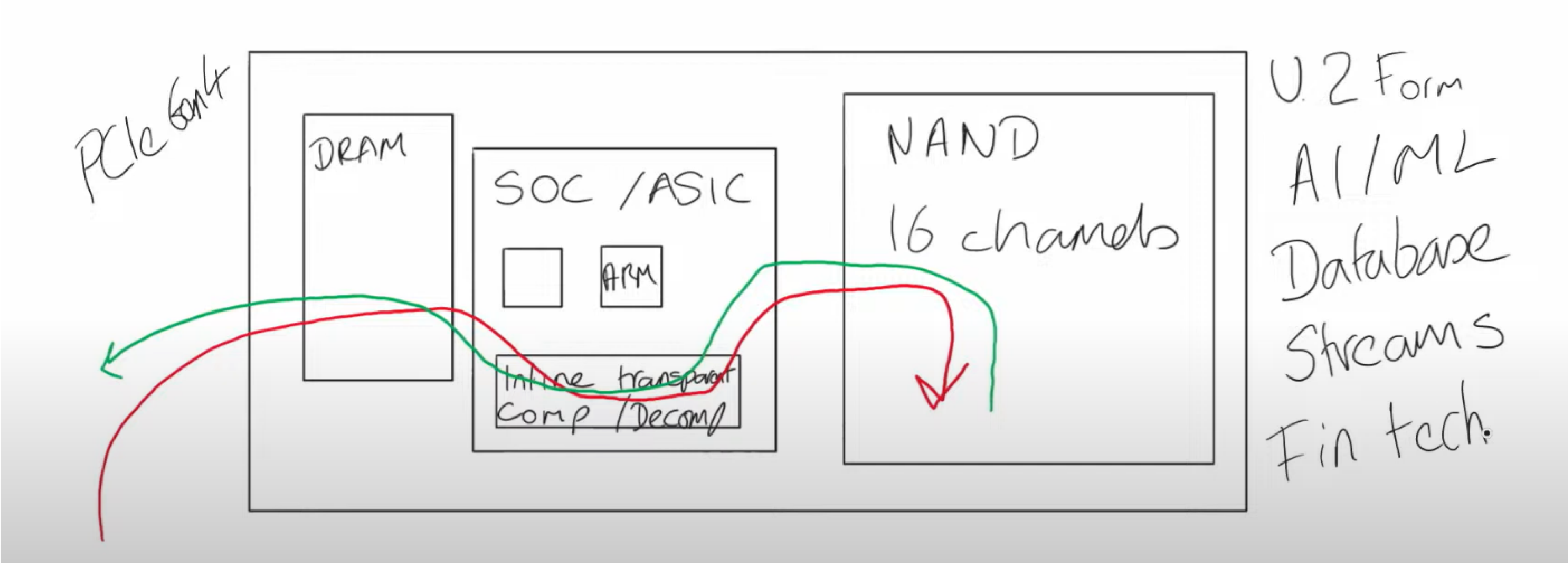
The heart of our system, and something that makes us a little different, is our ASIC, also known as a **System-on-a-Chip (SoC).** The SoC is custom designed by ScaleFlux, and built on several Arm cores and Hardware Accelerator Engines. The first function we implemented in computational storage was an **inline transparent compression and decompression** function decoupled from the host CPU.

To get data into the device, we use a PCIe Gen4 interface. It will work on older Gen 3 servers, but you won't quite see the performance that you would from GM, from a system with a Gen 4 bus. As we write data, it comes from the host into the DRAM, flows through, our ASIC gets compressed, and then gets written to the NAND device where it's persisted for data coming out.

Very similar. We read from those NAND channels that it gets hydrated or decompressed as it comes out of the device and then to the host. **Compression is always on**. You can use it in different ways, either having a regular sized namespace or doing something called **Expanded Namespace.**

Where can we use this CSD 3000?

We've seen customers use it on large datasets, mainly focused on **machine learning and artificial intelligence**. We have customers using it today in traditional structured and unstructured databases such as **NoSQL**.



High-level SSD Architecture

We also see customers using it for streaming data. These streams might be data from anything. For example, it might be market data from financial institutions interested in high-speed network packets manipulated and stored to calculate risk profiles as quickly as possible. In essence, the idea is simple. The benefit you get from compressing data on the storage device instead of elsewhere in the CPU or only on the network allows you to gain **additional performance** from your application.

2nd Order Effects

Another possible outcome is **increased license efficiency**. For example, if you're licensing an application such as **SQL Server** or **Oracle**, you could get more work done per license. Or, you could be particularly sensitive to rack power, thermal, or other considerations from a sustainability perspective. But using computational storage, especially with compression built into the storage device itself, allows you to be more efficient on many levels.

It would help if you looked at your application performance is bound. Are you looking at system resource constraints in DRAM, CPU, or just that your storage device or array can't keep up? That's the first indication that this could be a great fit.

Is your dataset compressible?

Most people don't know whether their data is compressible, and some believe it is entirely uncompressible. Uncompressible data is a possibility, but it's extremely rare in practice. ScaleFlux can provide you with a tool that allows you to assess the compressibility of your dataset in the applications you care about and very quickly give an indication of the degree of benefit you could see by deploying CSD 3000 in your architecture.

In addition, our device is a U.2 form factor device, meaning you need a server with a U.2 backplane; ideally, it should be Gen 4 PCIe. They do work in the earlier Gen 3. Still, you're going to limit performance, and the whole point of using a computational storage device is to **increase performance capacity and endurance**.

If you want to know more about how to realize performance gains, how to get that capacity utilization up to a very high level, or even extend your capacity namespace by two or three times, please get in touch with us.