A KAMINARIO WHITE PAPER

# The Importance of Variable Block Sizes in an All-Flash Array

November 2013

# The Importance of Variable Block Sizes in an All-Flash Array

Flash providers report on 4KB (or 8KB at best) block I/O in order to demonstrate performance. Why? Because that's the block size they support, and by definition the smaller the block size I/O the higher the IOPS for a given bandwidth – and most of these providers have very low bandwidth to begin with, so they can only achieve "flashy" IOPS numbers by using very small block sizes.

The unfortunate reality is that a 4KB I/O size rarely exists in the real world. It's significantly smaller than the typical I/O size in all leading databases across any environment. In fact, the average I/O size delivered is about 40KB, ten times bigger than the I/O in 4KB benchmarks. Also notable: in real world scenarios, I/O size varies based on the time of day and day of the week. This can range from as low as 8KB to as high as 128KB – or even higher. In particular, OLAP applications (e.g. Real Time Analytics) produce enormous I/O requests.

## Scale-Up Architectures and bandwidth limitations

As systems grow, and more loads are added to the system, the limited bandwidth that can be generated from a dual controller system becomes insufficient. Systems that include a maximum of one or two active controllers (which by definition makes them Scale-Up, NOT Scale-Out) are normally limited to a maximum bandwidth of 2 to 4GB/sec, since the controller is limiting the system bandwidth.

Even when they proclaim to support variable block sizes, these scale-up systems have bandwidth limitations that in some situations are even lower than legacy systems with HDDs – due to the serious bottleneck that is having only one or two controllers in the system that all traffic must go through.

## Scale-Out Architectures and fixed block sizes

Most scale-out flash arrays (Kaminario excluded) are architected around a constant 4KB internal block size. This means that when they receive I/O, it is split into multiple separate 4KB requests.

If an I/O request that is above 4KB is split internally into multiple flash I/Os, that adds to the latency increasing exponentially: CPU power on the array (to process all the splits), and latency of the flash media (10 writes for a 40 KB I/O vs one write), compound itself to increase latency exponentially, further reducing IOPS and limiting the bandwidth to very low numbers.

Although it is expected that IOPS drop to some extent with increases in I/O size, with all flash arrays the opposite should be true of served bandwidth; in other words, bandwidth should increase as I/O size increases – that's what happens with Kaminario – but if your flash array is splitting the requests, performance will truly tank as I/O requests get bigger: bandwidth will drop as I/O sizes get bigger, so IOPS will drop extremely quickly (in fact, they will drop exponentially).

Why? Latency is a the heart of all the other performance metrics – low latency is why flash performs so well – so the latency overhead caused by fixed block sizes has a compounding effect on IOPS and bandwidth. Here are the IOPS formulas for **flash arrays** (where the concept of seek time does not exist – seek time is a characteristic of HDDs):

$$\text{IOPS} = \frac{1}{\text{Transfer time of I/O + Compound latency of all block read/writes (in seconds)}}$$

$$\text{Bandwidth (KB/s)} = \frac{\text{Transfer Size (KB)}}{\text{Transfer time of I/O + Compound latency of all block read/writes (in seconds)}}$$

**Let's look at the 40 KB example.**

a) Let's assume a fixed transfer time of 10 microseconds for those 40 KB (which translates to about 4 GB/s, a fairly modest but realistic number).

b) Let's also assume latency for each back-end block read/write of 200 microseconds.

**With an array with 4 KB fixed blocks**, meaning 10 back-end transfers to get 40 KB, we will have:

$$\text{IOPS} = \frac{1}{0.00001 + 10 \times 0.0002} = \frac{.1}{0.00201} = 498 \text{ IOPS}$$

$$\text{Bandwidth} = \frac{40}{0.00201} = 20 \text{ MB/s}$$

Therefore, if a flash array has a small, fixed block size, that significantly limits the ability to utilize the bandwidth of the flash media for real loads that are on average 40KB I/O.

**With an array with variable block size**, meaning the 40 KB will get written in one go, we will have:

$$\text{IOPS} = \frac{1}{0.00001 + 0.0002} = \frac{.1}{0.00021} = 4{,}762 \text{ IOPS}$$

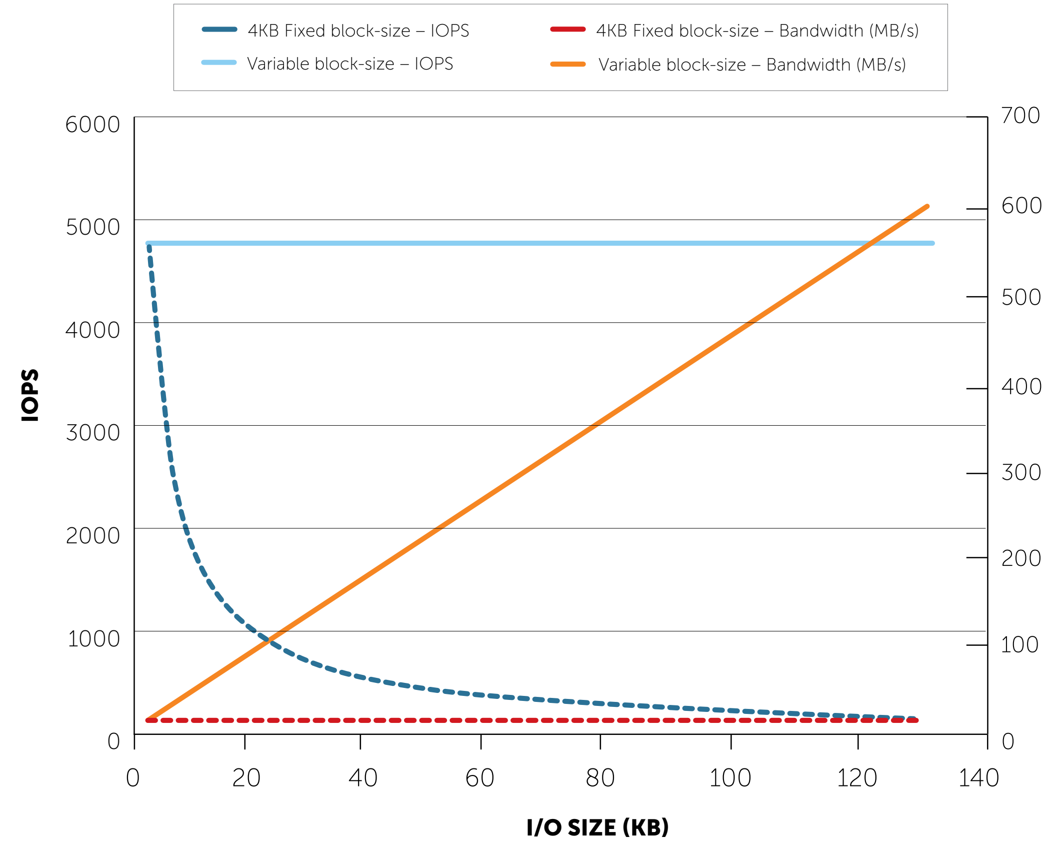
$$\text{Bandwidth} = \frac{40}{0.00021} = 190 \text{ MB/s}$$

So it's clear how much better performance having a variable block size can bring in this case.

Now, if after looking at the above numbers you are wondering "*Well, why not have a large fixed block size then?*". Well if we have, say, a 128 KB block size, applications can end up spending significant system bandwidth or increased latency to bring in large

quantities of data that were not needed in the first place.
This chart illustrates the difference quite vividly:

FIXED BLOCK SIZE VS. VARIABLE BLOCK SIZE



With a variable block size you get the no-compromises best of both worlds: lowest possible latency and highest possible IOPS and bandwidth in all situations.

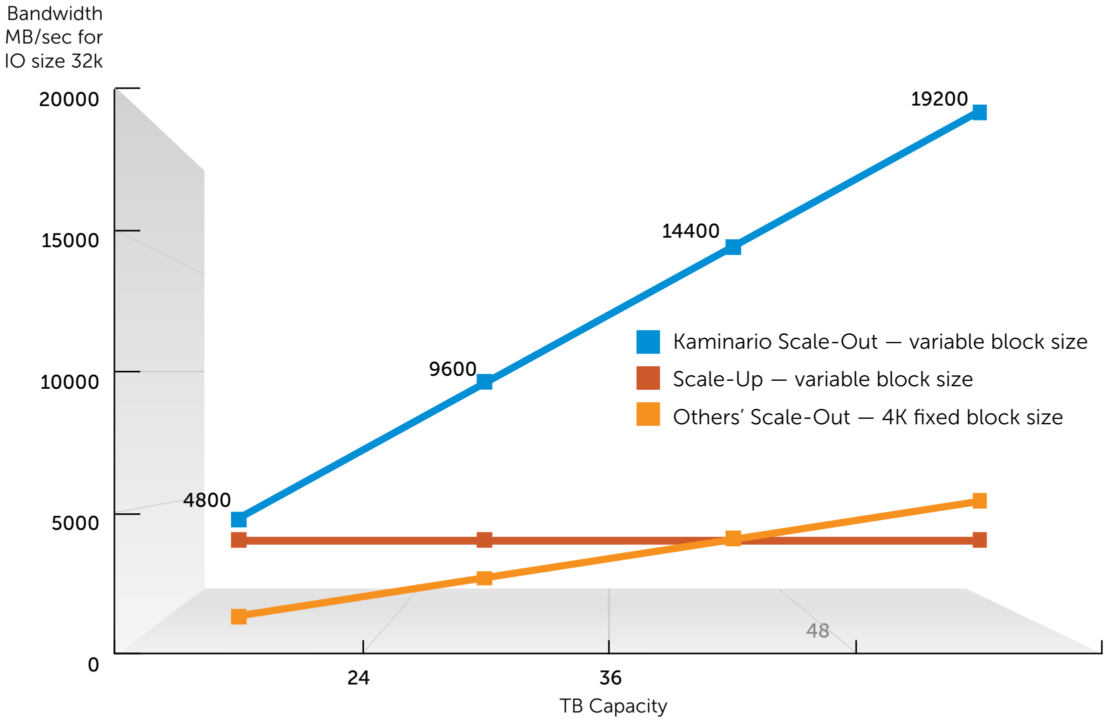
## The Kaminario Difference

Kaminario K2 is a Scale-out all-flash array built to support mixed workloads that combine high IOPS and bandwidth with low latency. K2 optimizes the I/O layout and handling to meet the actual customer needs by marrying a variable block size algorithm – optimized for any I/O size – with a true scale-out architecture. A K2 system bandwidth can scale from a few GB/sec to more than 20GB/sec while running real mixed customer loads.

What's more, K2 snapshots match the I/O size from the application. This means that a write operation does not over allocate capacity, and a read operation does not need to fetch more data or split the read I/O into many small requests. This combination significantly improves the read bandwidth performance.

More specifically, snapshots are managed in 4KB I/O granularity, while the data is stored on the SSD media continuously in 4KB to 128k increments. This means that the system is automatically optimized to the actual load, and enables efficient metadata (MD) management without impacting the read throughput of real customer I/O (which is normally much greater than 4KB size). This is why Kaminario's bandwidth performance, no matter how you slice it, is unsurpassed by any other flash solution on the market. The graph below demonstrates scale-up approaches and other scale-out approaches as compared to Kaminario's approach:

**MIXED DATABASE WORKLOAD—32KB IO, 90% READ / 10% WRITE**



To summarize, real world I/Os are typically much larger than 4KB. So, when you're comparing vendors, make sure these requirements are on your checklist:

1. Do they support variable block sizes?

2. Do they have a scale-out Architecture?