It's important to strike the right balance w.r.t the number of storage server processes per disk.

If your system has a lot of data being written (e.g. during initial load), it's generally good to have more than a single storage server process per disk. That's because the SSD storage engine issues one write IO at a time, while the disks can handle much more, and only perform at their peak with much more than that. With a lot of writes coming in and not efficiently pushing them on to the disks, the storage queue on the storage server starts to grow and becomes a throughput bottleneck when it reaches its limit. By adding more storage server processes per disk, they will be able to issue more IOs to the disk concurrently which will translate into using more of the disks capacity and that helps with performance.

To be very precise the SSD engine does often have an IO queue depth of 1 but it is due to waiting on reads. Each set or clear must do a BTree seek for each key (or range endpoint) and any time those seeks encounter a non-cached page they must wait for it. Then they can modify the page and "write" it but the write is buffered and not sent to disk until commit time, the idea being that the page may be modified again.

The SSD engine does do up to 64 get() or getRange() operations at once, but that queue is independent of the Writer thread which must do its own reads serially (using the shared cache) as part of its update path.