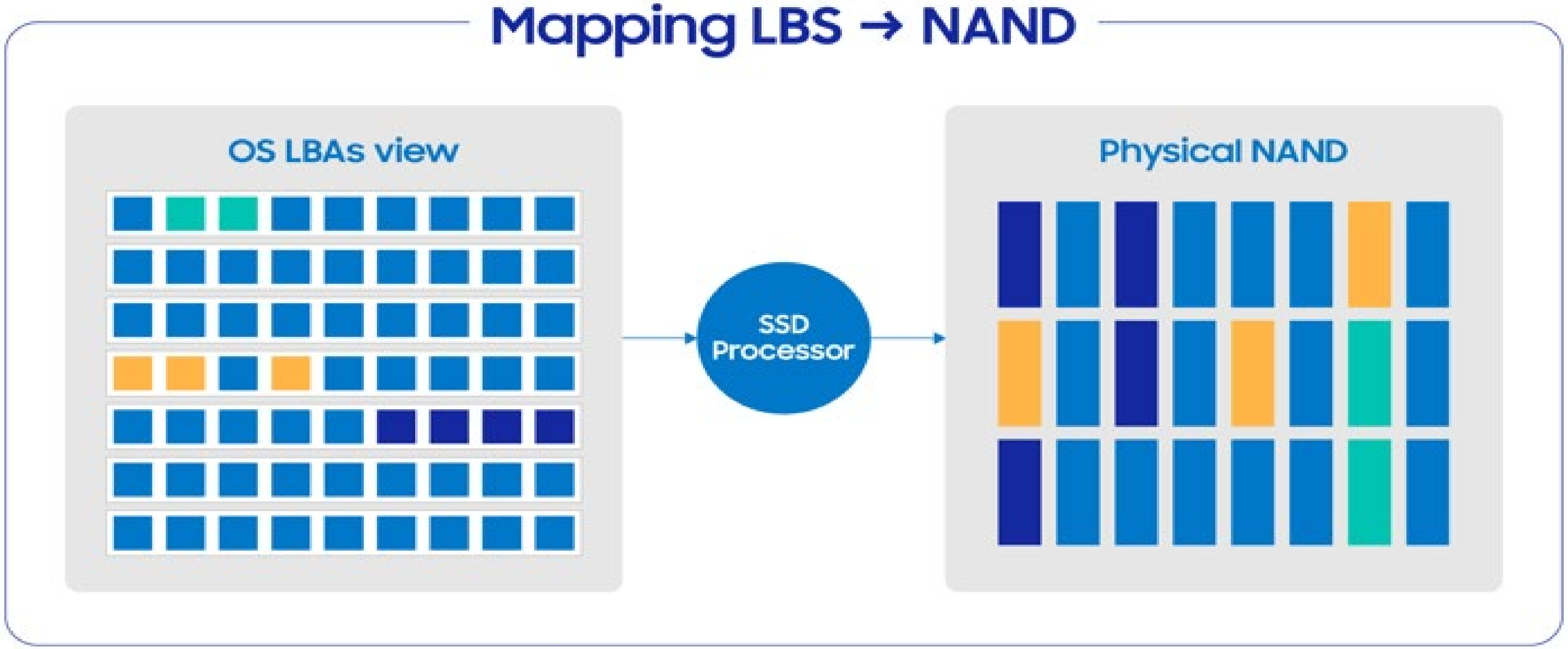
Seamless Adoption of QLC HC-SSDs with LBS

August 6, 2025

Luis Chamberlain mcgrof.c@samsung.com

Principal Engineer

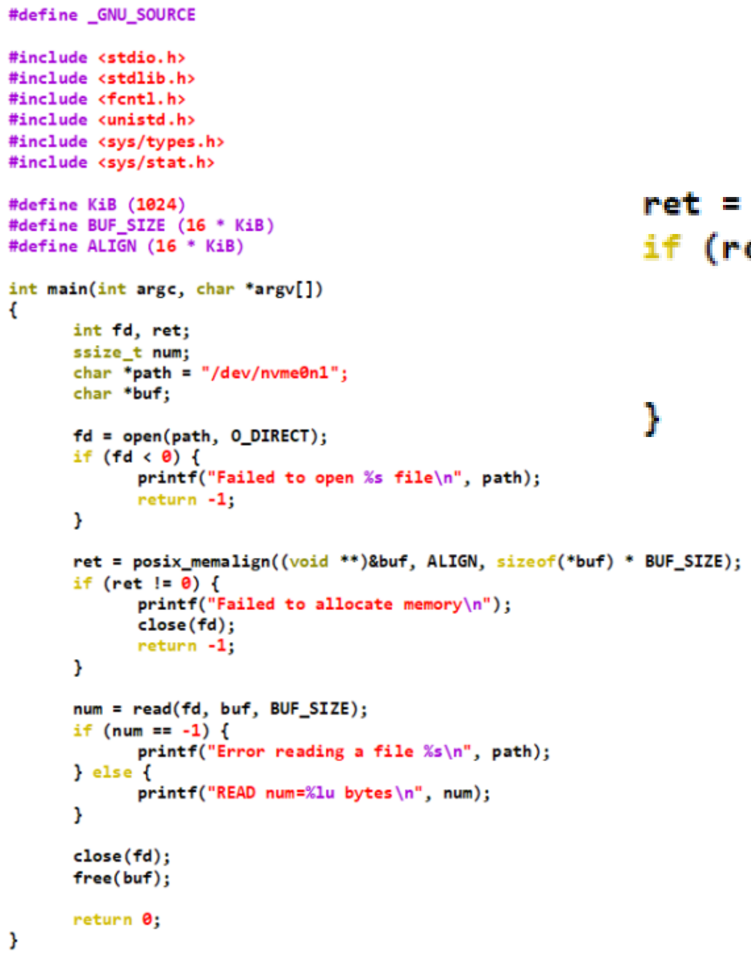
IU mapping table at a glance: an internal structure



The DRAM implications for HC-SSDs

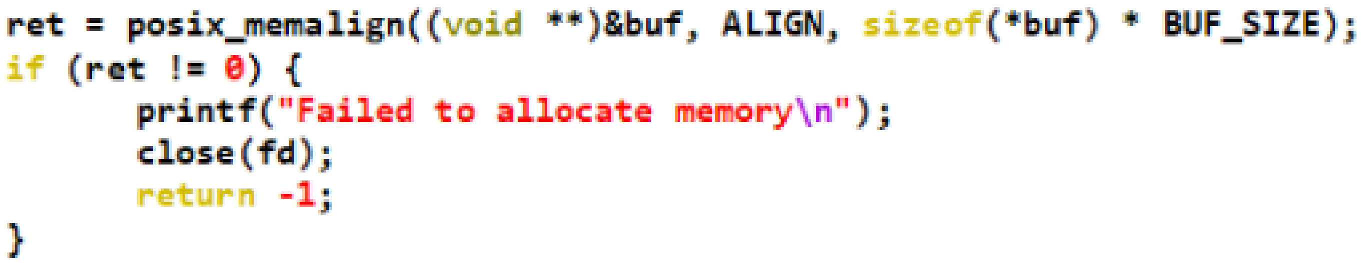
Drive capacity	DRAM required with 4k IU	DRAM required with 16k IU	Savings by using 16k IU
7.67 TiB	7.67 GiB	1.92 GiB	5.75 GiB
15.34 TiB	15.34 GiB	3.84 GiB	11.05 GiB
30.68 TiB	30.68 GiB	7.67 GiB	23.01 GiB
61.44 TiB	61.44 GiB	15.36 GiB	46.09 GiB

Userspace recommendations for large IU without LBS



```
#define _GNU_SOURCE

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

#define KiB (1024)
#define BUF_SIZE (16 * KiB)
#define ALIGN (16 * KiB)

int main(int argc, char *argv[])
{
        int fd, ret;
        ssize_t num;
        char *path = "/dev/nvme0n1";
        char *buf;

        fd = open(path, O_DIRECT);
        if (fd < 0) {
                printf("Failed to open %s file\n", path);
                return -1;
        }

        ret = posix_memalign((void **)&buf, ALIGN, sizeof(*buf) * BUF_SIZE);
        if (ret != 0) {
                printf("Failed to allocate memory\n");
                close(fd);
                return -1;
        }

        num = read(fd, buf, BUF_SIZE);
        if (num == -1) {
                printf("Error reading a file %s\n", path);
        } else {
                printf("READ num=%lu bytes\n", num);
        }

        close(fd);
        free(buf);

        return 0;
}
```

Figure 4. Code snippet showing example implementation of optimal I/O: direct I/O performed on raw block device that adheres to IU size and alignment recommendations (in this example IU is 16KiB in size)



```
ret = posix_memalign((void **)&buf, ALIGN, sizeof(*buf) * BUF_SIZE);
if (ret != 0) {
        printf("Failed to allocate memory\n");
        close(fd);
        return -1;
}
```

- Requires userspace applications to be modified
- Each new IU increase implicates new modifications
- Not a suitable ecosystem choice

Alternatives to requiring userspace changes

Lay of the land of what options we have

1. LBA format change → likely not possible
   - 4 KiB LBA format introduced 1998
   - 4 KiB LBA Native format 4kn in 2010
   - Slow adoption
   - Requires new Protection Information protection changes
   - This alternative is not as ideal
2. Operating Systems R&D innovation: The Large Block Size moonshot goal

The Large Block Sizes moonshot goal

Goal: respect IU alignment using OS filesystem primitives

- Primitives:
  - `mkfs.xfs -f -b size=16k`
  - `mkfs.xfs -f -b size=16k -s size=16k`
- Would be ideal but people had tried for 16 years

The Large Block Sizes moonshot goal

Goal: respect IU alignment using filesystem primitives

- Primitives:
  - `mkfs.xfs -f -b size=16k` **Merged on v6.12**
  - `mkfs.xfs -f -b size=16k -s size=16k` **Merged on v6.15**

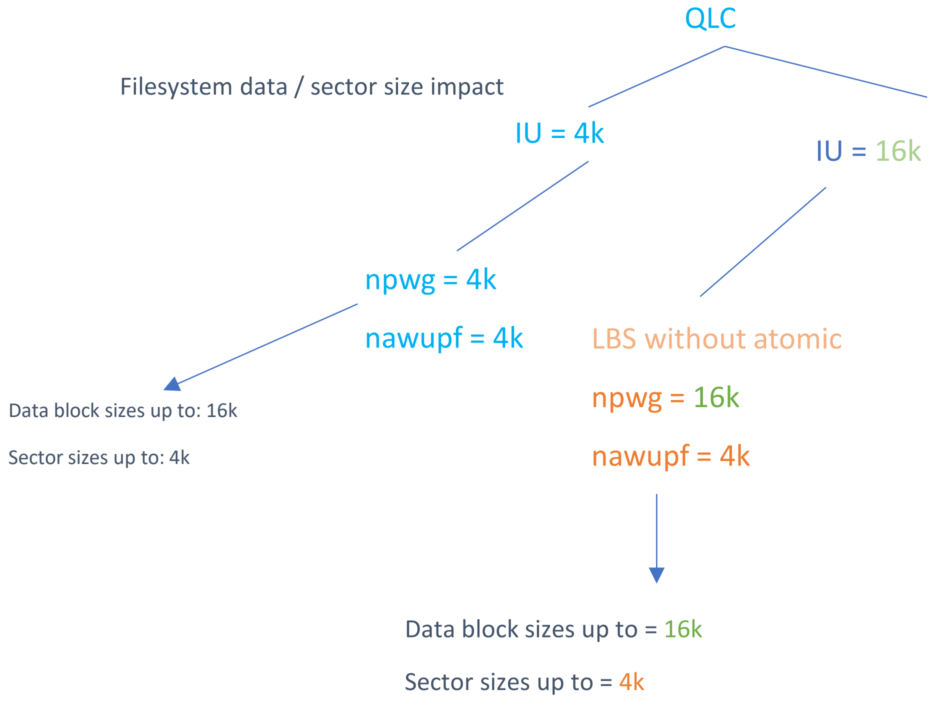
Done

It was just an Operating Systems Filesystems and Memory Management Problem

What does LBS mean in practice?

For filesystems, and the QLC High-Capacity SSD market?

NVMe QLC with IU=NPWG=16k



QLC

Filesystem data / sector size impact

IU = 4k

IU = 16k

npwg = 4k

nawupf = 4k

Data block sizes up to: 16k

Sector sizes up to: 4k

LBS without atomic

npwg = 16k

nawupf = 4k

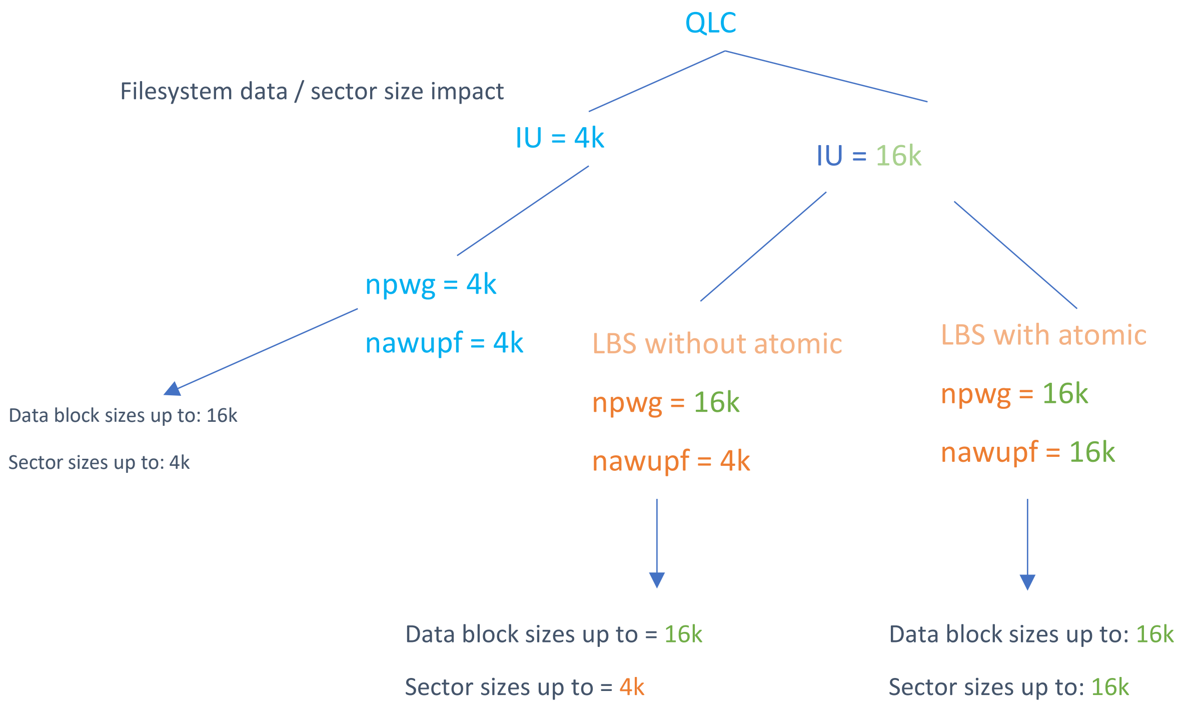
Data block sizes up to = 16k

Sector sizes up to = 4k

- `mkfs.xfs -f -b size=16k`

the Future of Memory and Storage

NVMe QLC with IU=NPWG=16k and NAWUPF=16k



QLC

Filesystem data / sector size impact

IU = 4k

IU = 16k

npwg = 4k

nawupf = 4k

Data block sizes up to: 16k

Sector sizes up to: 4k

LBS without atomic

npwg = 16k

nawupf = 4k

Data block sizes up to = 16k

Sector sizes up to = 4k

LBS with atomic

npwg = 16k

nawupf = 16k

Data block sizes up to: 16k

Sector sizes up to: 16k

- `mkfs.xfs -f -b size=16k`
- `mkfs.xfs -f -b size=16k -s size=16k`

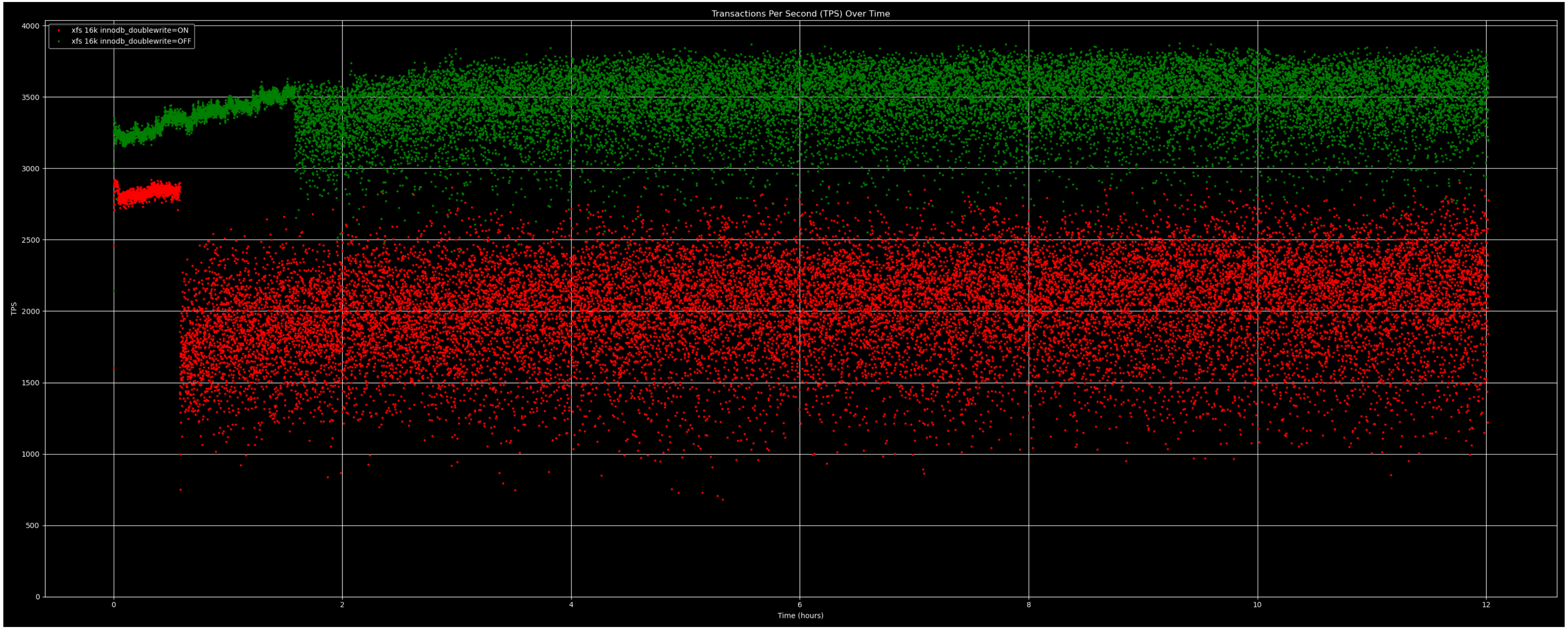
But are atomics useful?

Hyperscalers have been supporting large atomics for 6 years now

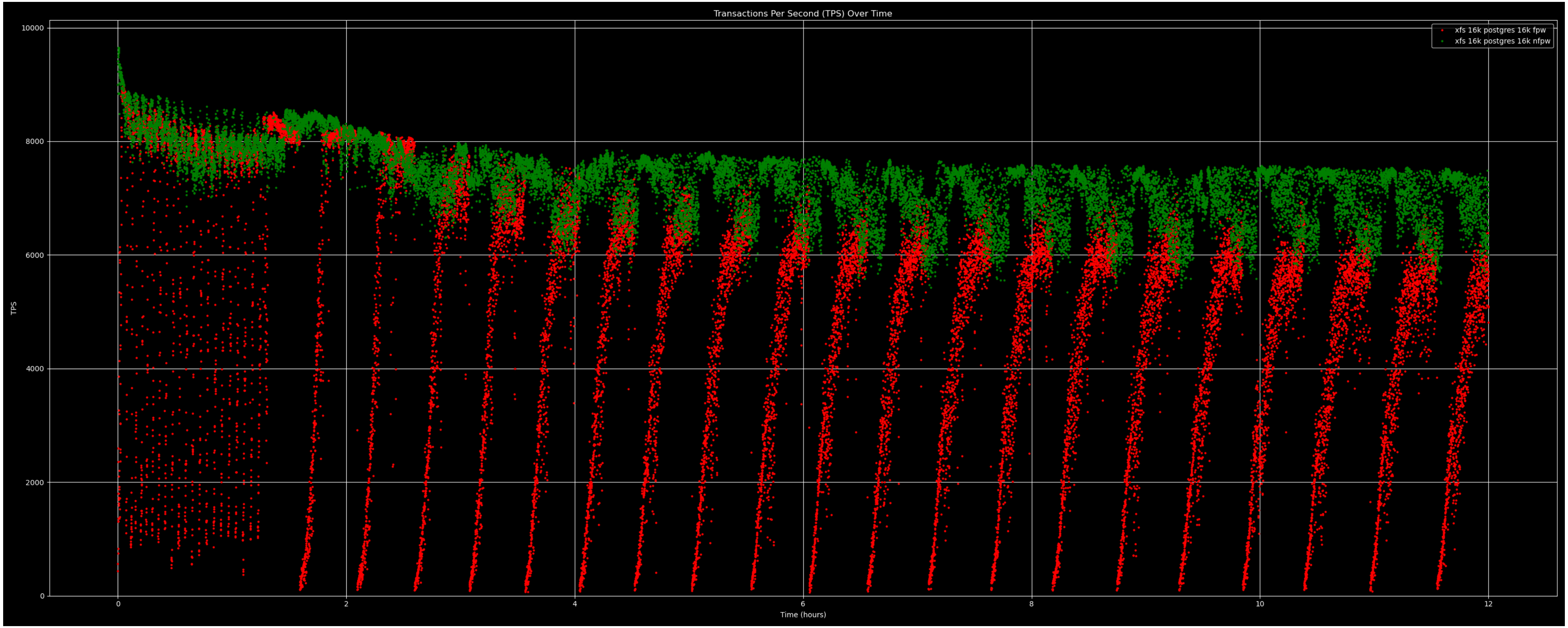
Let’s test on **AWS i4i.4xlarge** as a public baseline on TLC

Leverage kdevops sysbench for reproducibility

Additional gains of large AWUPF: Disabling MySQL innodb_double_write_buffer 12 hour MySQL run → you can reproduce with kdevops



Additional gains of large AWUPF: Disabling PostgreSQL full_page_writes
12 hour run → you can reproduce with kdevops



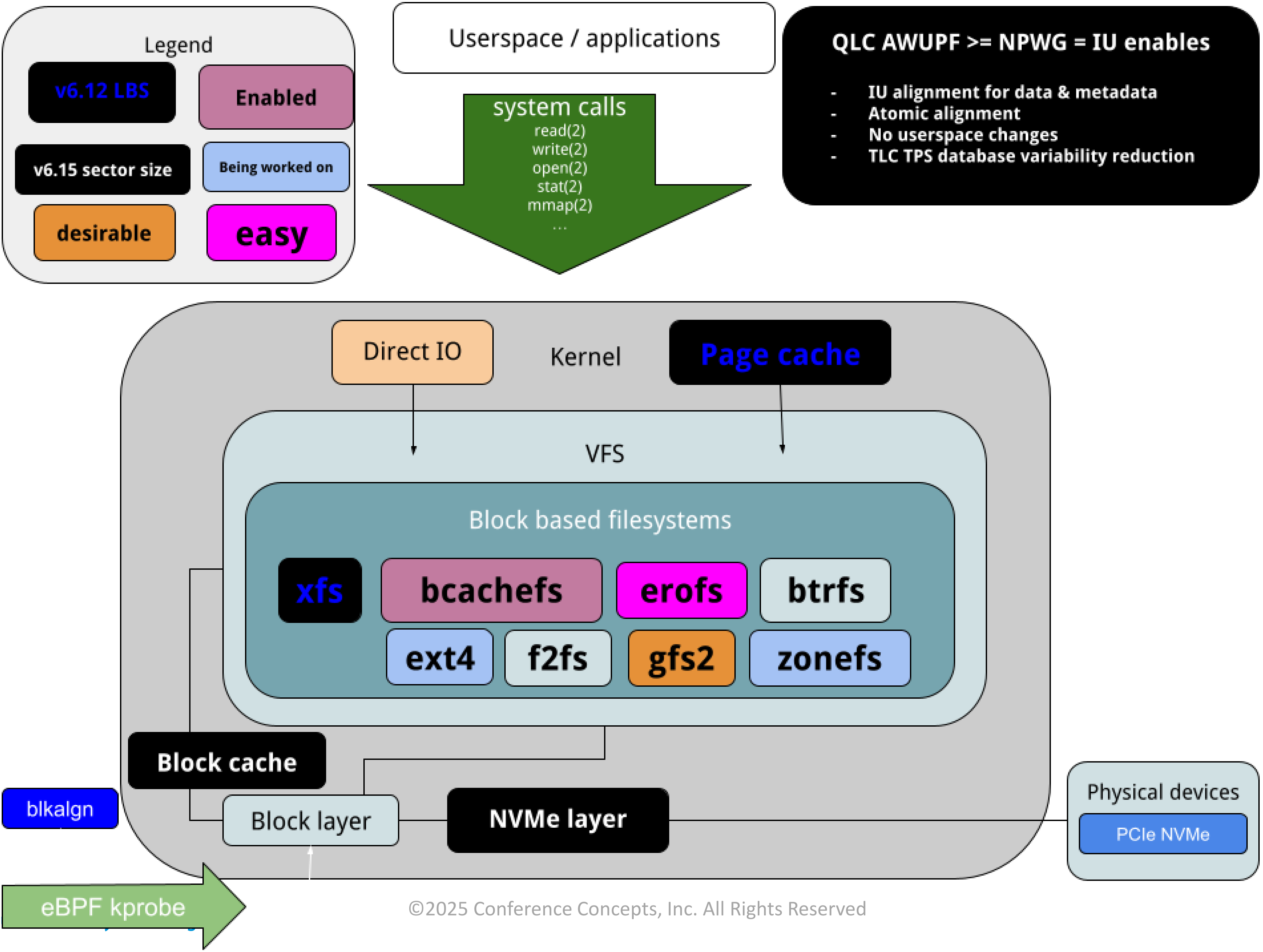
FMS the Future of Memory and Storage

But are atomics useful?

Yes

An empirical evaluation of large atomics shows they are

Useful even for TLC drives then



QLC AWUPF >= NPWG = IU enables

- IU alignment for data & metadata
- Atomic alignment
- No userspace changes
- TLC TPS database variability reduction

Legend

v6.12 LBS

Enabled

v6.15 sector size

Being worked on

desirable

easy

Userspace / applications

system calls
read(2)
write(2)
open(2)
stat(2)
mmap(2)
…

Kernel

Direct IO

Page cache

VFS

Block based filesystems

xfs

bcachefs

erofs

btrfs

ext4

f2fs

gfs2

zonefs

Block cache

Block layer

NVMe layer

blkalgn

eBPF kprobe

Physical devices

PCIe NVMe