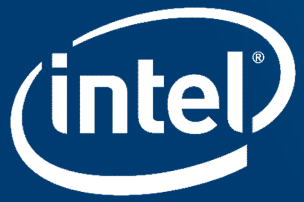


SPDK BLOBSTORE

Ben Walker
Data Center Group
Intel Corporation

Notices and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit **http://www.intel.com/performance**.

AGENDA

- INTRODUCTION
- USE CASES
- DESIGN
- BENCHMARKS

INTRODUCTION

LOTS OF APPLICATIONS WANT TO USE SPDK...

BUT THEY AREN'T DESIGNED TO DIRECTLY USE THE BLOCK DEVICE

WHAT DOES A FILESYSTEM DO?

PARTITIONS

PERMISSIONS

CACHING

RAID

SNAPSHOTS

ACCESS TIMES

BYTE GRANULARITY

SPARSE ALLOCATION

DIRECTORIES

TRIM

CHECKSUMS

I/O SCHEDULING

WHAT CAN SPDK DO TO HELP?

LET'S BUILD SOME NEW COMPONENTS!

USE CASES

WHAT SORT OF APPLICATION BENEFITS FROM SPDK?

Lots of I/O

Latency Sensitive

- SAN? Database? Cache?

We picked two use cases:

- RocksDB
- Dynamic Block Allocation

ROCKSDB

- Log-structured merge tree
- Written in C++, Open Source
- Pluggable storage backend
- Broadly adopted
- Recommends XFS

Makes minimal use of XFS

- Directory structure
- I/O pattern
- Minimal caching needs

NO OTHER FILE SYSTEM FEATURES REQUIRED!

DESIGN

GLOSSARY OF TERMS

- File: Array of bytes
  - Mutable, Resizable
  - String name
- Object: Array Of bytes
  - Immutable, replaceable
  - String name
- Page – 4 KiB

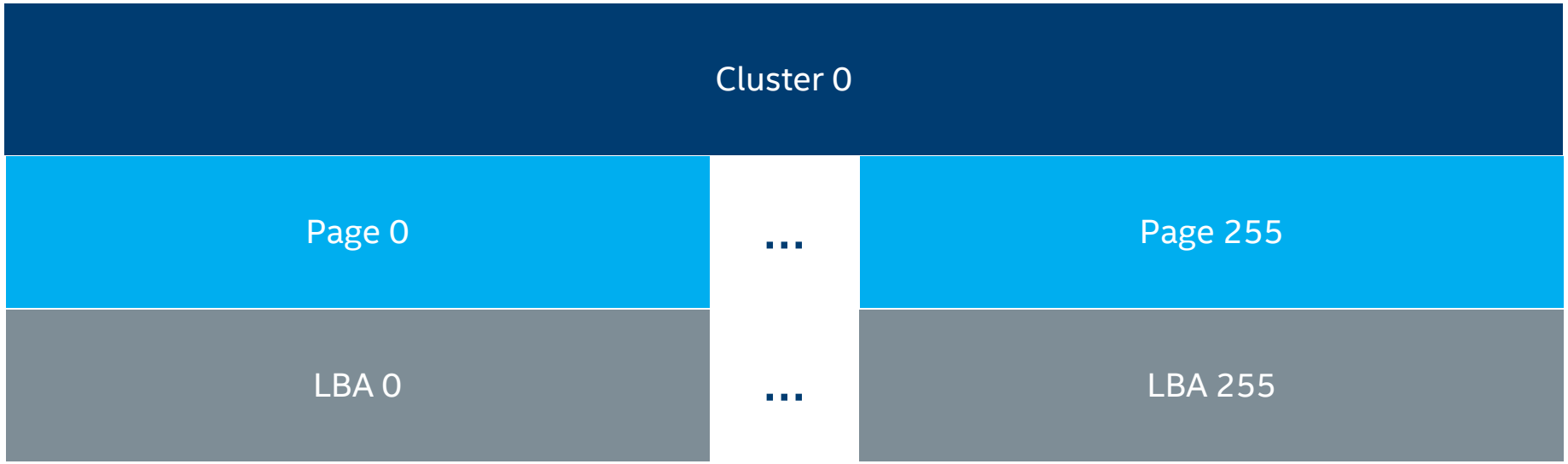
DESIGN GOALS

- Simple and efficient
- Design for fast storage media
- Support file & object-like semantics

BLOBSTORE BASICS

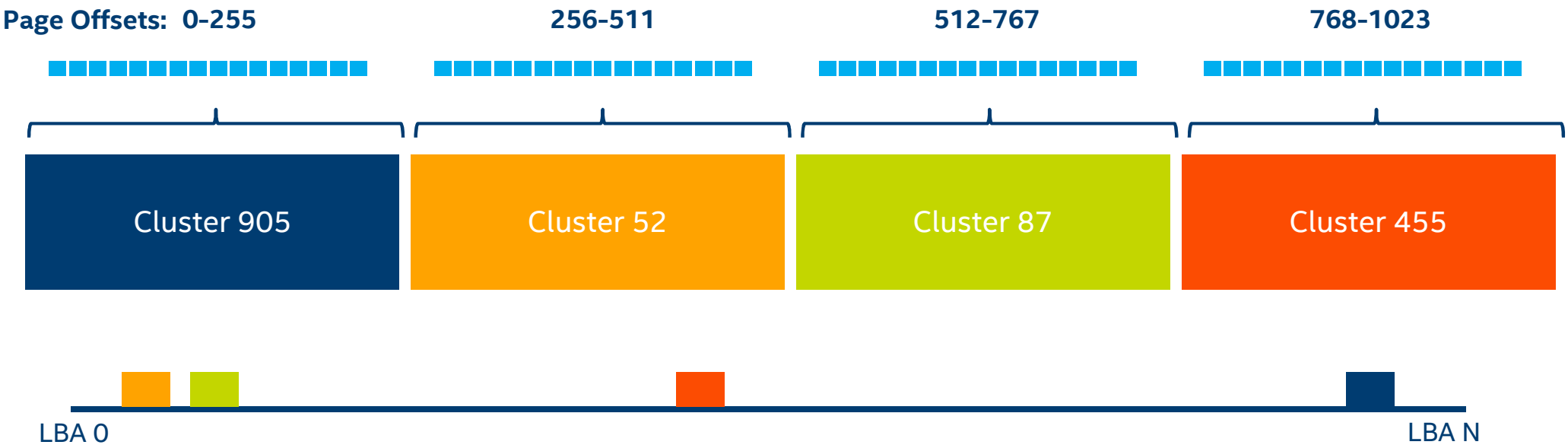
- The user interacts with chunks of data called blobs
  - Array of pages
  - Mutable, resizable
  - ID
- Asynchronous
  - No blocking, queueing, or waiting
- Fully parallel operations
  - No locks in I/O path

BLOBSTORE SPACE ALLOCATION



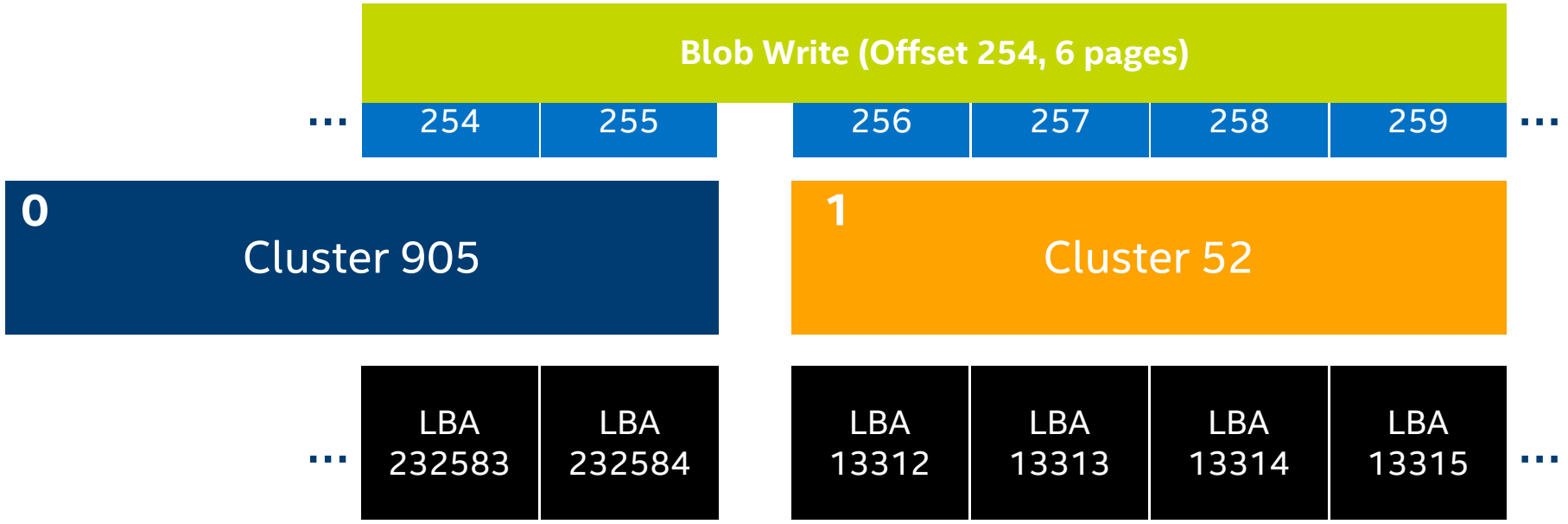
BLOBSTORE DESIGN

Blob: **array of pages** implemented as an ordered **list of clusters**:



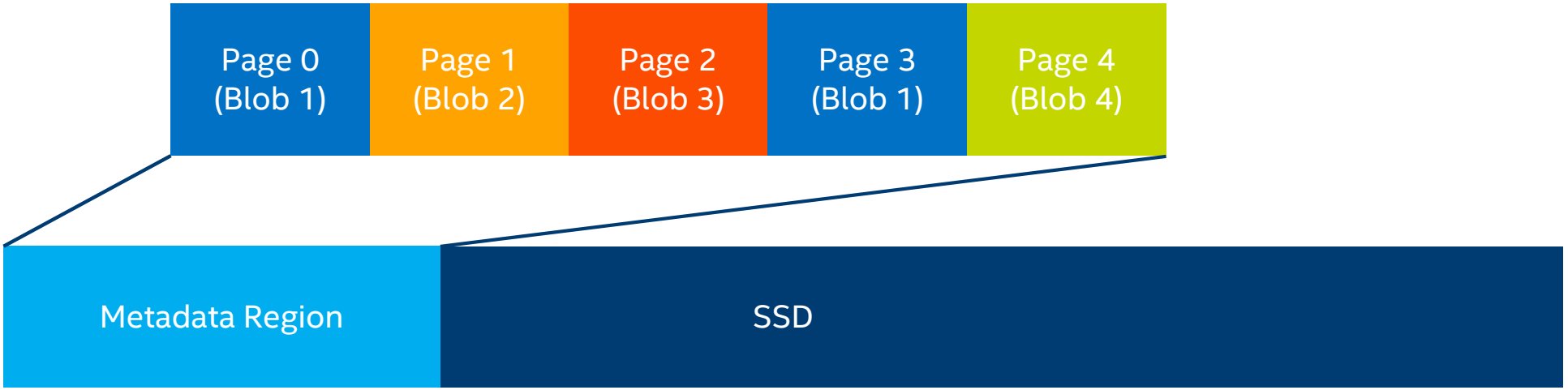
BLOBSTORE SAMPLE I/O

Blobs are read/written by specifying a relative **page offset** and a **page count**



BLOBSTORE METADATA

- Metadata is stored in **pages** in a reserved region
- Metadata pages are **not shared** between blobs
- A blob may have multiple pages of metadata



BLOBSTORE API

- **open, close, read, write, sync, resize**
- Asynchronous, callback-driven
- Read/write in units of pages, space allocation in clusters

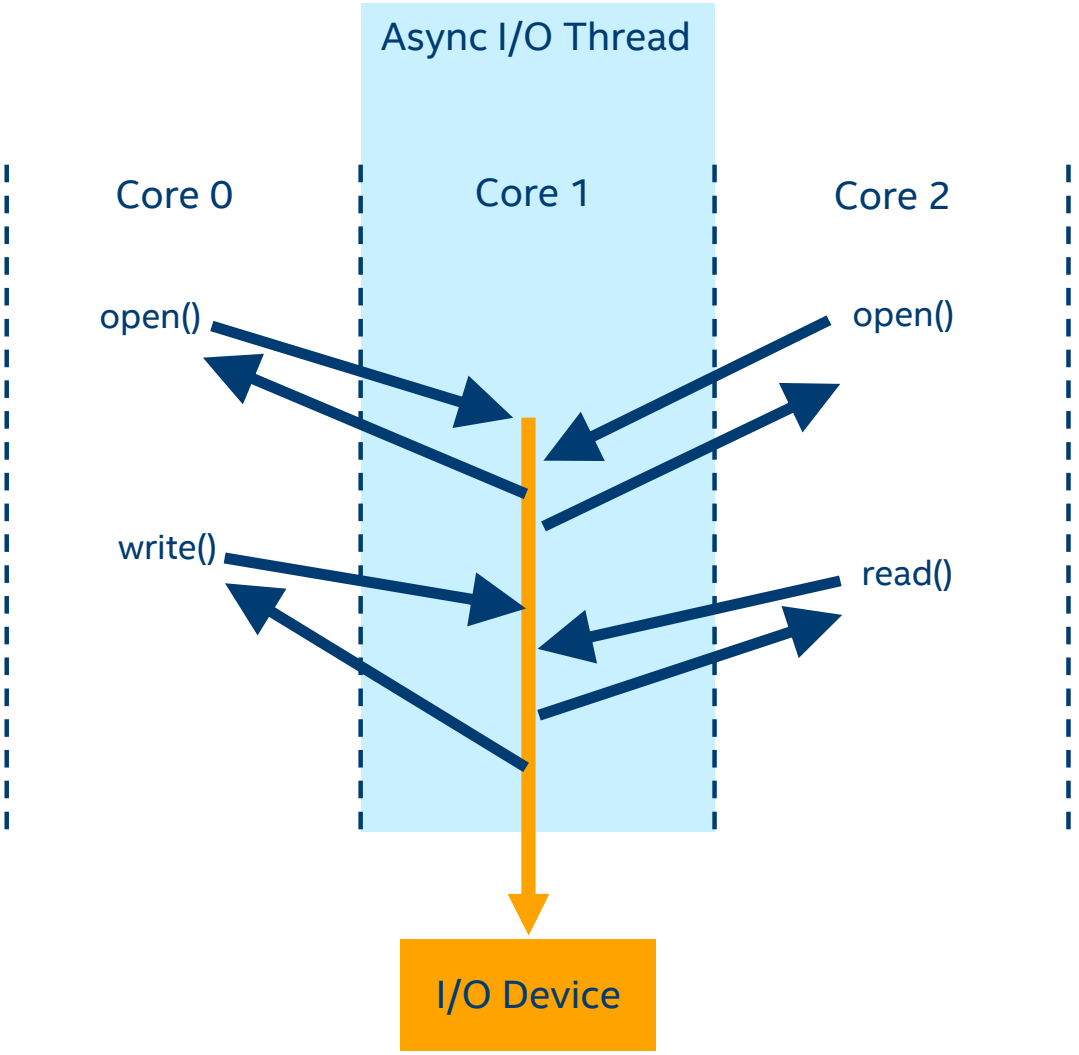
- Data is **direct**
- Metadata is **cached**
- Minimal support for xattrs

INDEPENDENT OF BLOBFS

BLOBFS DESIGN

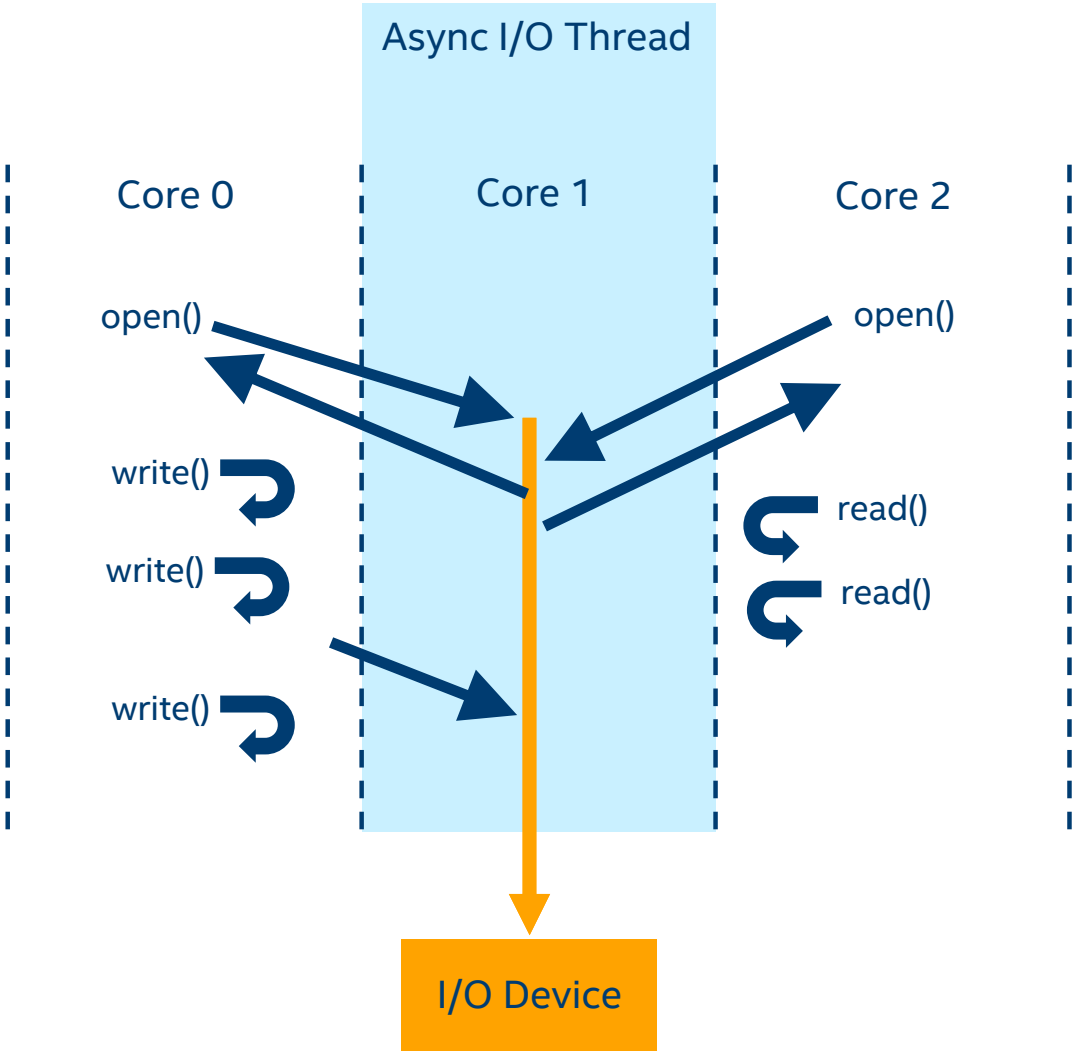
- Layered on Blobstore
- User interacts with **files**
- Data can be **cached**
- **Synchronous API***

* Asynchronous API possible



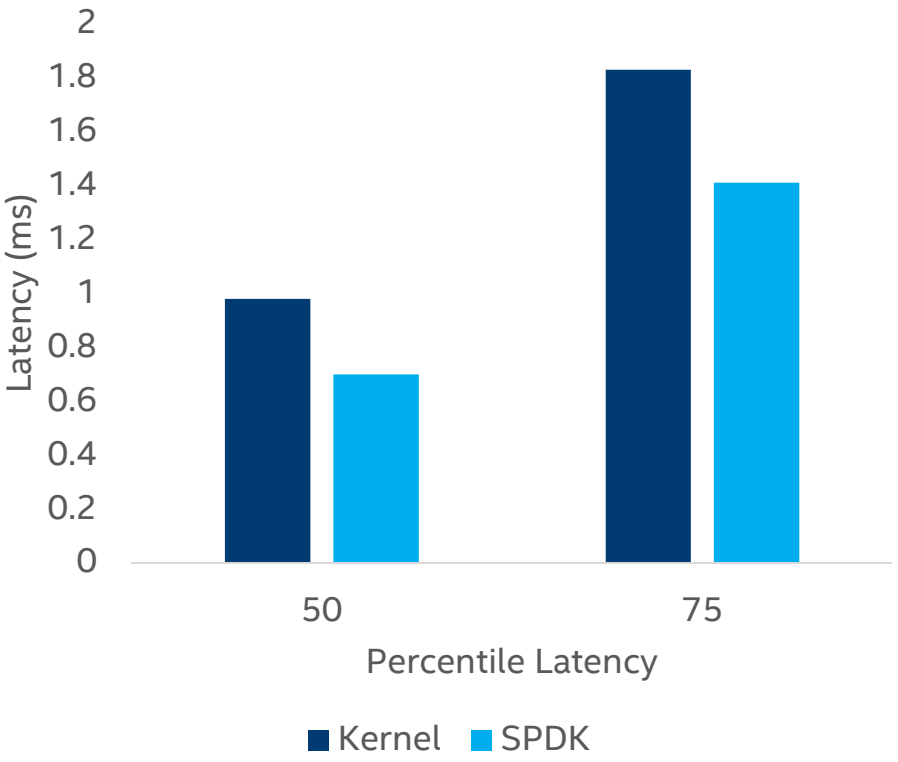
BLOBFS CACHING

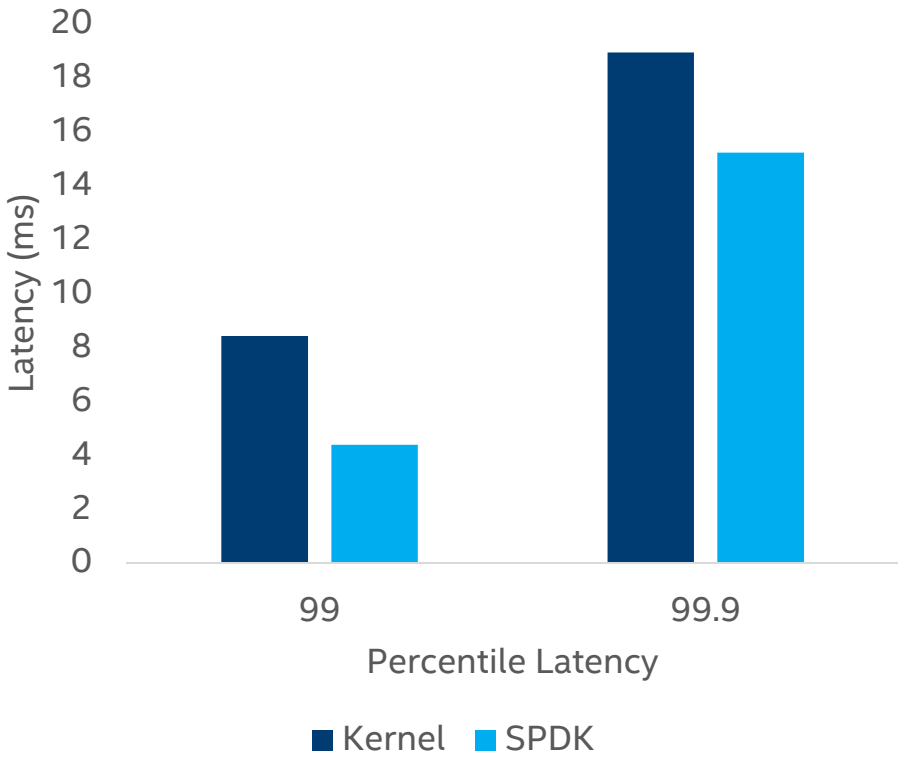
- Not a general purpose page cache
  - Read ahead
  - Sequential write buffering
  - All other access bypasses cache



BENCHMARKS

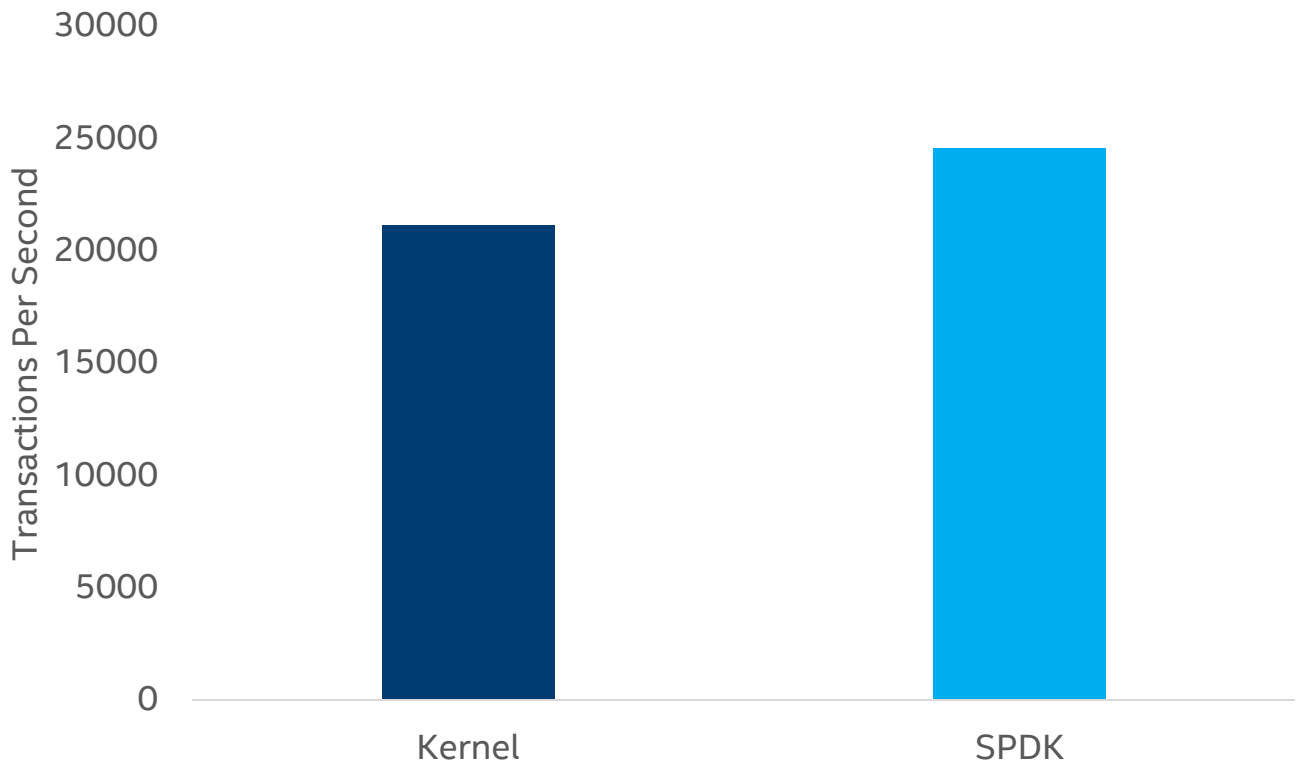
BENCHMARK: DB_BENCH READ/WRITE LATENCY





System Configuration: 2x Intel® Xeon® E5-2699v3, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled, 8x 8GB DDR4 2133 MT/s, 1 DIMM per channel, Fedora* Linux 25, Linux kernel 4.10.8, Intel® P3700 NVMe SSD (800GB), FW 8DV101H0, SPDK 17.03, DPDK 17.02, RocksDB 5.1.2

BENCHMARK: DB_BENCH READ/WRITE THROUGHPUT



System Configuration: 2x Intel® Xeon® E5-2699v3, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled, 8x 8GB DDR4 2133 MT/s, 1 DIMM per channel, Fedora* Linux 25, Linux kernel 4.10.8, Intel® P3700 NVMe SSD (800GB), FW 8DV101H0, SPDK 17.03, DPDK 17.02, RocksDB 5.1.2

NEXT STEPS

NEXT STEPS

- Major API clarifications
- More & better benchmarking
- Use blobstore as a dynamic partitioner (bdev)
- BlobFS caching strategy is RocksDB-centric
- Asynchronous BlobFS API
- Sparse allocation of blobs
- More open source application integration?

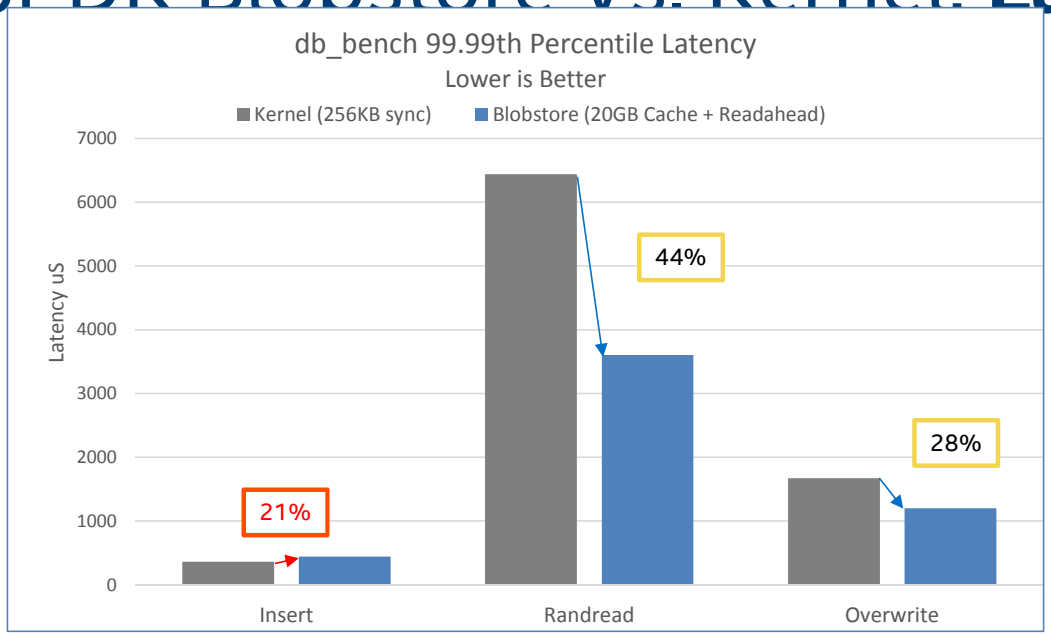
EFFICIENCY

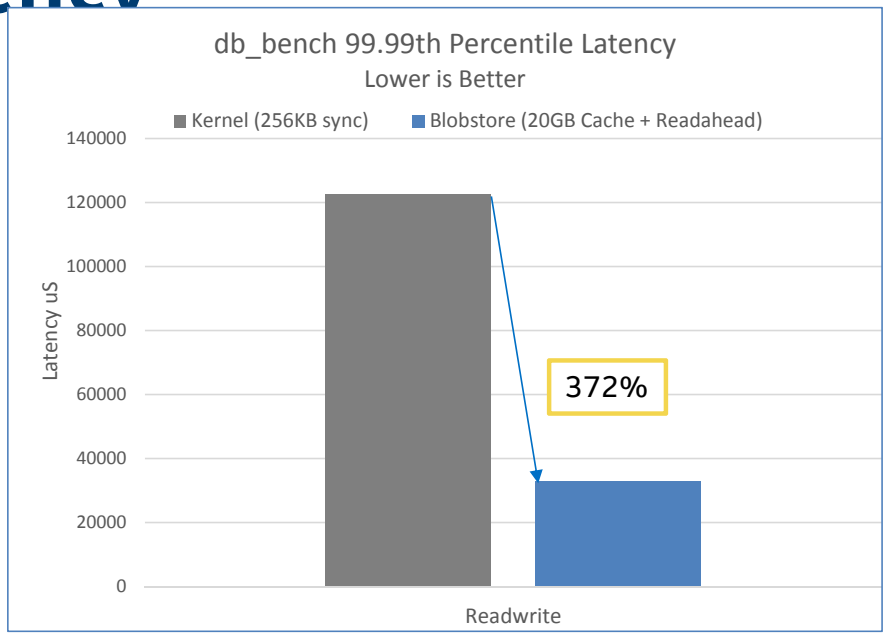
SIMPLICITY

FLEXIBILITY


intel®
experience
what's inside™

SPDK Blobstore Vs. Kernel: **Latency**

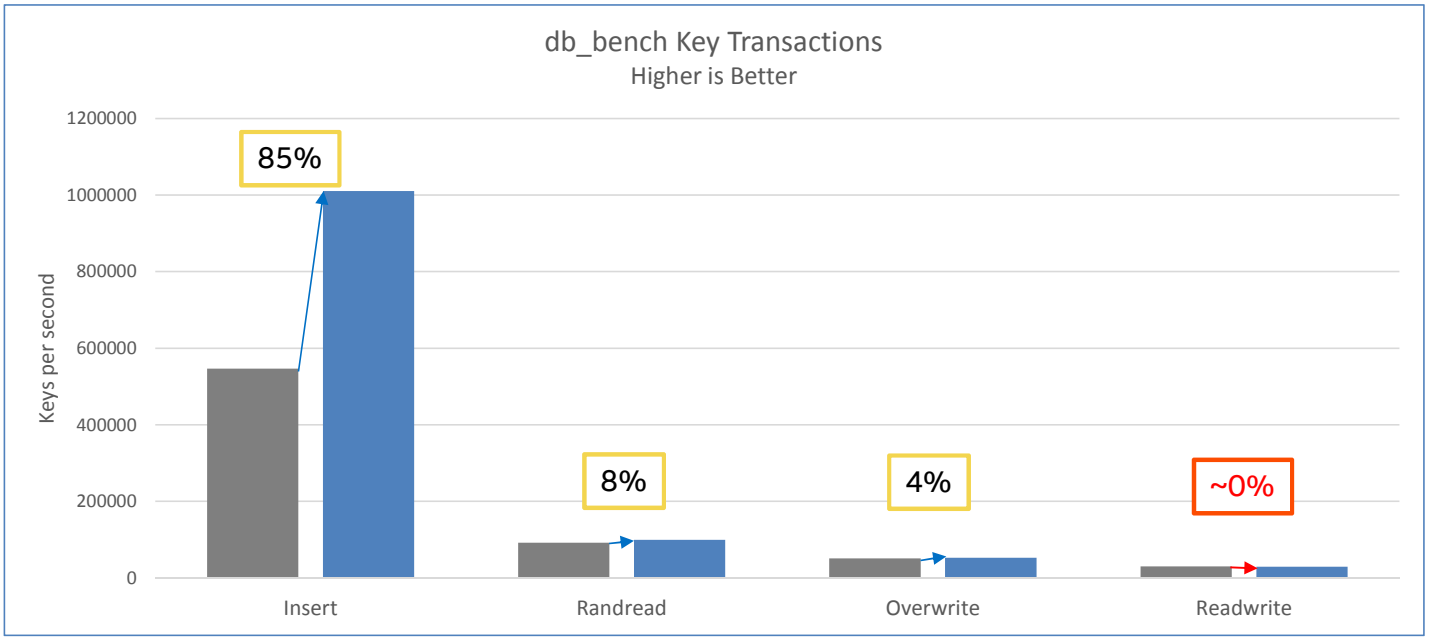




SPDK Blobstore reduces tail latency by 3.7X

System Configuration: 2x Intel® Xeon® E5-2699v3, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled, 8x 8GB DDR4 2133 MT/s, 1 DIMM per channel, Fedora* Linux 25, Linux kernel 4.10.8, Intel® P3700 NVMe SSD (800GB), FW 8DV101H0, SPDK 17.03, DPDK 17.03, RocksDB 5.1.2

SPDK Blobstore Vs. Kernel: **Transactions Per Second**



System Configuration: 2x Intel® Xeon® E5-2699v3, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled, 8x 8GB DDR4 2133 MT/s, 1 DIMM per channel, Fedora* Linux 25, Linux kernel 4.10.8, Intel® P3700 NVMe SSD (800GB), FW 8DV101H0, SPDK 17.03, DPDK 17.03, RocksDB 5.1.2