
# Open Source Data Reduction for High Performance Flash Storage

Louis Imershein

Principal Product Manager, Red Hat

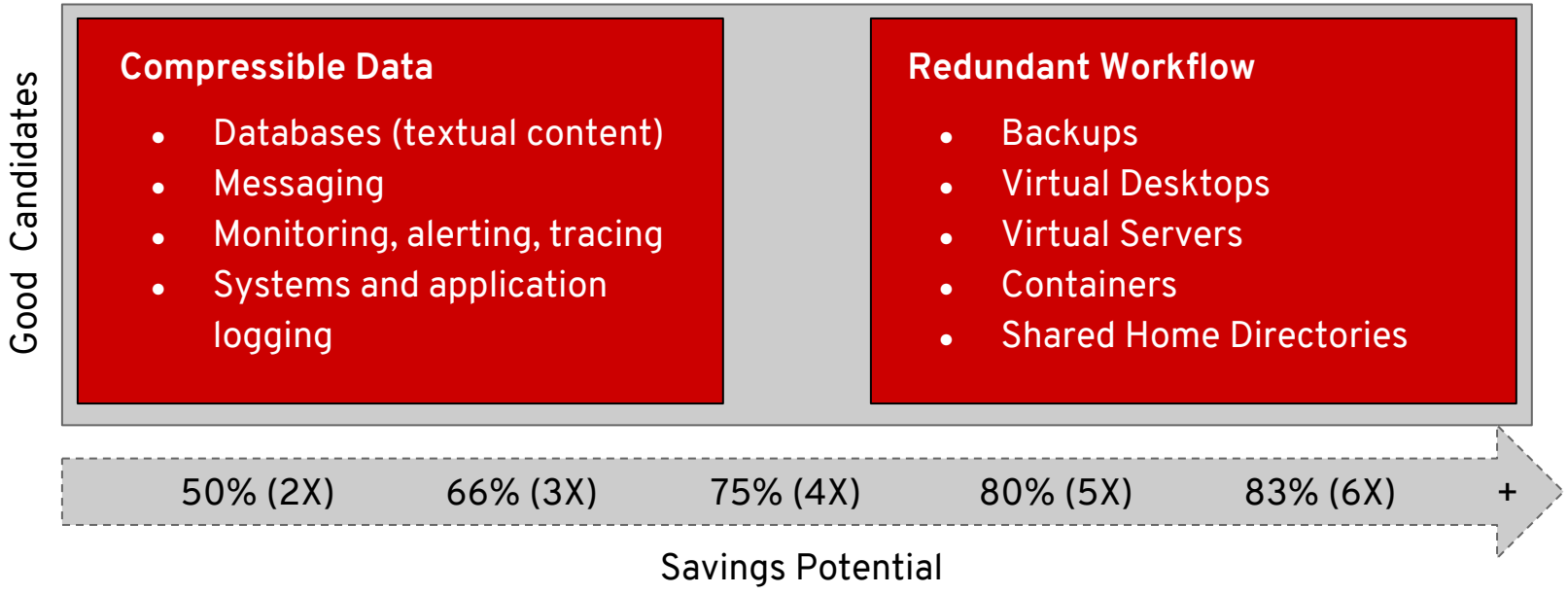

# Data Reduction Overview

- Data reduction incorporates two space saving technologies, compression and deduplication
- The goal of data reduction is to drive down the effective cost of storage by reducing the data footprint
- To do that, data reduction has to use processing and memory resources

Flash Memory Summit

# How Much Can We Save?

Data reduction savings are dependent on data type and workflow




# Flash and Data Reduction

- Challenges to developing data reduction solutions center around:
  - resource overhead
  - performance requirements
- Two data reduction targets: **Backup** and **Primary**
- Primary storage data reduction solutions are very good at using flash storage to offload meta-data from DRAM, one example is VDO


# What is VDO?

- VDO is a Linux device mapper target that provides software data reduction services at the  block level: deduplication and compression
- As a block device, VDO can be used underneath block, file or obejct storage
- VDO is purpose-built to leverage the capabilities of low-latency storage (Flash, 3D XPoint, etc…) to control resource costs
- VDO is Linux component first released with Red Hat Enterprise Linux 7.5 as an out-of-tree kernel module, available for several Linux distros today
- Open Source and GPL as of October 2017! (~165k LOC) https://github.com/dm-vdo


# Simple to install and configure

- Install packages (or clone and build github repo)
- Create VDO volume:
  `# vdo create --name=vdo1 --device=/dev/sdb`
- Monitor available physical space:
  `$ vdostats vdo1`
- Monitor available logical space:
  `$ df`
- Grow logical storage pool size:
  `# vdo growLogical --name=vdo1 --vdoLogicalSize=20TB`

**Or just use the Cockpit user interface (http://github.com/cockpit)**

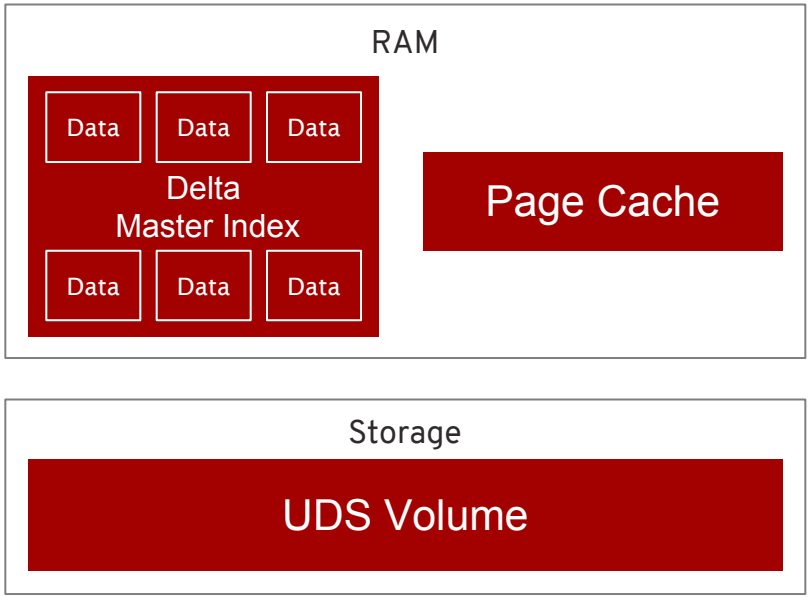

# Data Reduction Has Three Problems

1. Identifying duplicate items across enormous data sets
2. Mapping many small logical chunks to physical locations
3. Providing low latency access to reduced data

VDO includes two kernel modules that solve different problems:

- **uds** (universal deduplication service) - solves problem #1
- **kvdo** (kernel virtual data optimizer) - solves problems #2 and #3

# UDS: Fast Efficient Indexing




Fixed-size LRU Cache

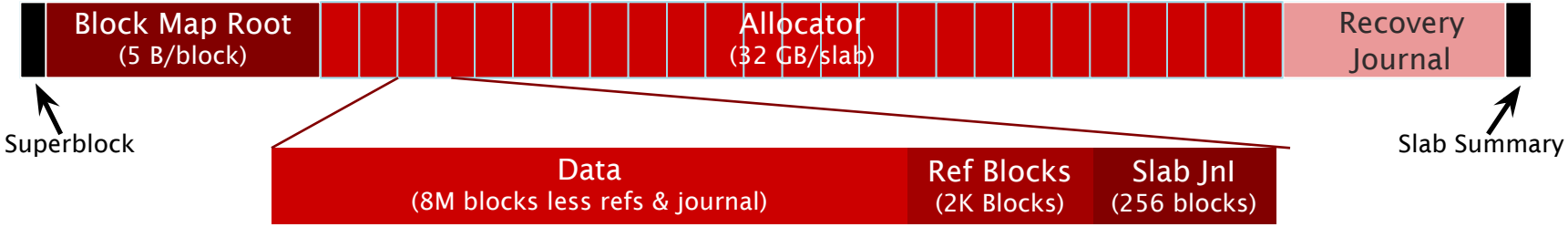
## Delta Master Index

- Quickly detects new duplicate items
- Unique compressed form uses only 3 bytes per active entry (representing 64 bytes of data)
- Sparse indexing allows 10x media coverage
- 5 usec average latency

## UDS Volume

- Log-structured database of key-value pairs
- Organized for data locality
- Maximum of 2 reads to find a record when not cached
- 64 KB pages cached for fast lookup
- >98% cache hit rate in small memory cache


# KVDO: Data Structures




## Write Amortization

- Map entries and refcounts journaled
- Data space divided into slabs for fast recovery and parallelism
- Unique data never re-written
- Explicit refcount management eliminates garbage collection

## Shortcut Processing

- Up to 2,000 I/O requests may be queued for write
- Logical and physical request maps allow requests to be satisfied against pending I/O
- Avoids read-read, read-write, write-read and write-write hazards

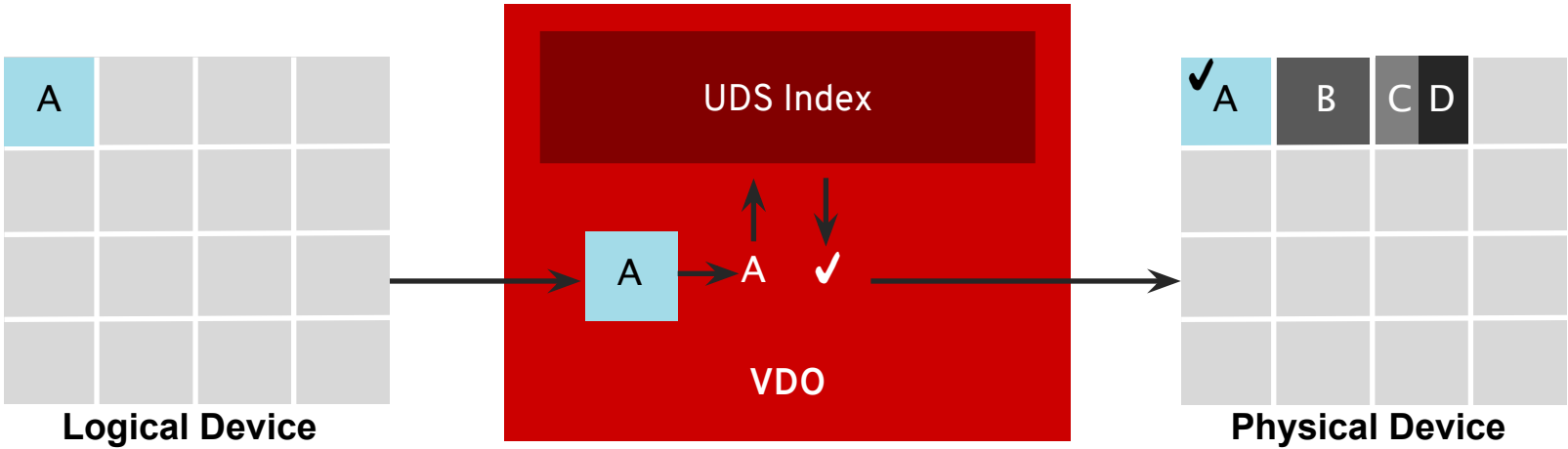

# Fast Hashing and Flash Storage

- VDO uses fast, non-cryptographic hashes (MurmurHash3)
- Read verify of existing block used to verify duplicate
- Resource overhead shifted off the CPU to flash storage

Flash Memory Summit

# VDO Compression

Only compress once - index hashes are on uncompressed block

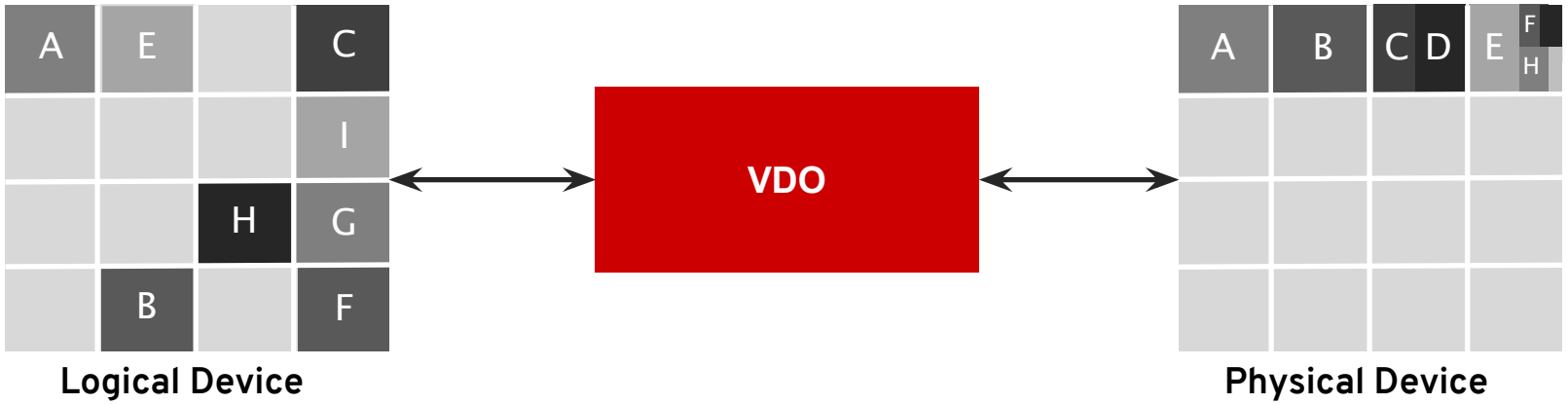



If the new block (“A”) matches against the index, only metadata is written.
UDS index can identify duplicates at >200k blocks/second/core.

Flash Memory Summit

# VDO: Compression

Compress fast (LZ4) and leverage multi-core systems

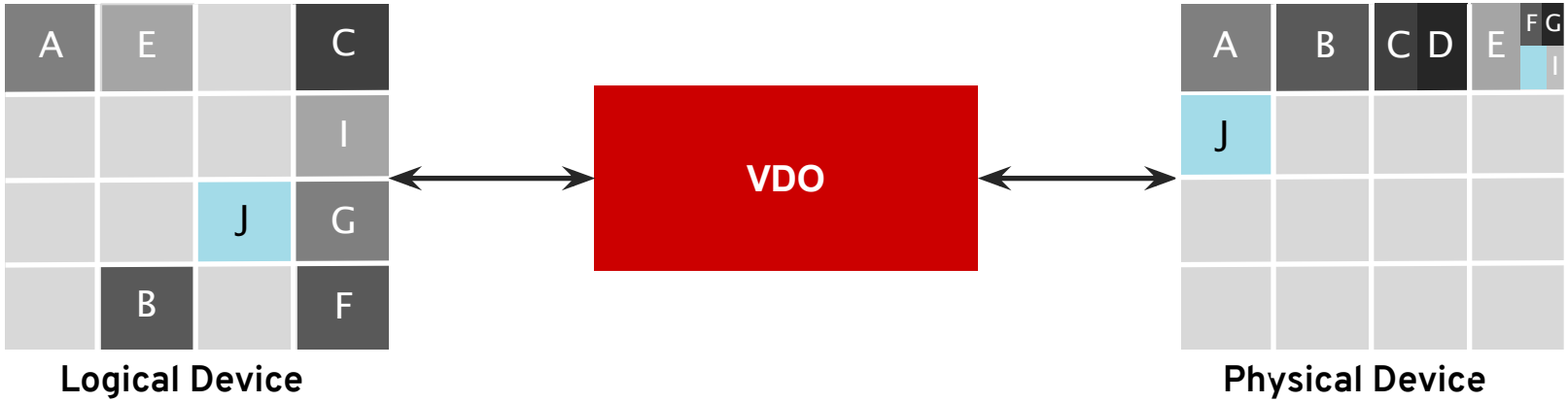



If the new block is unique, it is added to the index and compressed.
Up to 14 compressed blocks can be stored a single physical block.

Flash Memory Summit

# VDO: Garbage Collection

Track blocks and references in real time, without garbage collection




If a logical block is overwritten with unique data, new space is found.
If no more copies of the old block exist, its space is immediately available.
Compressed blocks are free when all fragments are unreferenced.


# VDO: Resources and Tuning

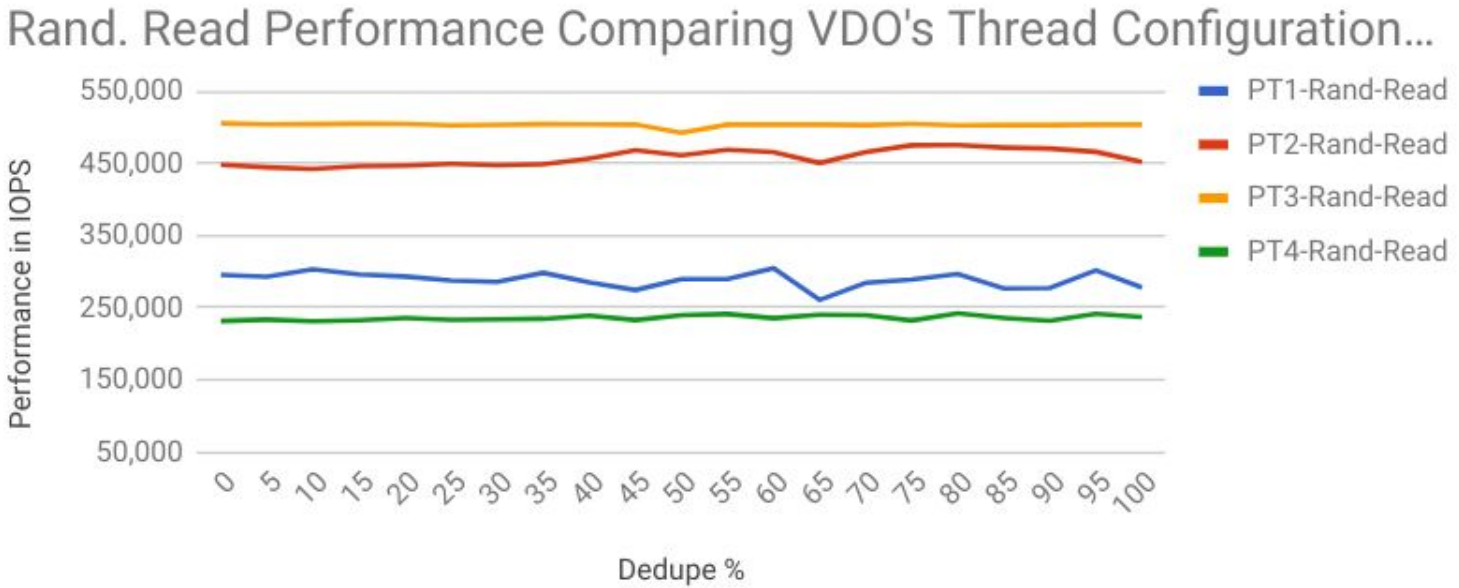
## Resource Requirements:

- RAM: 500 MB + 268 MB per TB of physical storage
- CPU: 40K-50K IOPS per core, depends on CPU and clockspeed

## Performance Tunables:

- Cached blockmap - size of logical-to-physical map in memory vs on media (default is 128MB. 128MB can cover 100GB of logical data)
- Thread settings - various settings, control how VDO scales across CPU cores

# Thread Configuration Matters




**CPU**: Intel Xeon Gold 6128 @ 3.4 Ghz, 6 core
**OS**: Red Hat Enterprise Linux 7.5

**Storage**: Intel Optane 280 GB
**Filesystem:** Xfs


# Interested in VDO?

**How to get VDO:**

- VDO was initially available in Red Hat Enterprise Linux 7.5
- Also now included in Fedora Copr, CentOS 7.5 and Oracle Linux 7 Update 5
- RPM Package Names:
  - **vdo** - User space utilities
  - **kmod-kvdo** - Kernel modules for uds and kvdo
- From source via: http://github.com/dm-vdo

**How to get involved:**

- VDO mailing list: **https://www.redhat.com/mailman/listinfo/vdo-devel**
  - Patches and discussions welcomed always
- VDO is openly available: **https://github.com/dm-vdo**
  - Email vdo-devel@redhat.com